Павлидис Т.

АЛГОРИТМЫ МАШИННОЙ ГРАФИКИ И ОБРАБОТКИ ИЗОБРАЖЕНИЙ

СОДЕРЖАНИЕ

Глава 1 Введение

1.1. Машинная графика, обработка изображений и распознавание образов

1.2. Виды данных, представленных в форме изображений

1.2.1. Класс 1: тоновые и цветные изображения

1.2.2. Класс 2: двухуровневые или представляемые в нескольких «цветах» изображения

1.2.3. Класс 3: непрерывные кривые и линии

1.2.4. Класс 4: точки или многоугольники

1.3. Ввод изображений

1.4. Устройства визуального отображения

1.5. Векторная графика

1.6, растровая графика

1.7. Общепринятые элементарные команды воспроизведения графических объектов

1.8. Сравнение векторных и растровых графических устройств

1.9. Программа-редактор изображений

1.10. Преобразования изображений

1.11. Система записи алгоритмов

1.12. Несколько замечаний относительно сложности

1.13. Библиографические замечания

1.14. Задачи

Глава 2 Дискретизация тоновых изображений

2.1. Введение

2.2. Преобразования Фурье и преобразования других видов

2.3. Выборка

2.3.1. Выборка в одномерном случае

2.3.2. Выборка в двухмерном случае

2.4. Искажения изображений вследствие недостаточно высокой частоты выборки

2.5. Квантование

2.6. Библиографические замечания

2.7. Задачи

Приложение 2.а. Быстрое преобразование Фурье

Глава 3 Обработка тоновых изображений
3.1. Введение
3.2. Гистограммы и их выравнивание
3.3. Матрицы совместной встречаемости уровней серого тона
3.4. Линейная фильтрация изображений
3.5. Нелинейная фильтрация изображений
3.5.1. Направленные фильтры
3.5.2. Составные фильтры
3.5.3. Фильтры приближения функций
3.6. Библиографические замечания
3.7. Задачи

Глава 4 Сегментация
4.1. Введение
4.2. Разделение по порогу
4.3. Обнаружение границ
4.4. Сегментация на основе выделения областей путем наращивания
4.4.1. Сегментация на основе среднего уровня яркости
4.4.2. Другие критерии однородности области
4.5. Библиографические замечания
4.6. Задачи

Глава 5 Проекции
5.1. Введение
5.2. Введение в методы восстановления изображений
5.3. Описание одного класса алгоритмов восстановления изображений
5.4. Применение проекций для анализа формы объектов
5.5. Библиографические замечания
5.6. Задачи
Приложение 5.а. Простая программа восстановления изображений

Глава 6 Структуры данных
6.1. Введение
6.2. Алгоритмы обхода графов
6.3. Страничная организация памяти
6.4. Пирамиды или тетрарные деревья
6.4.1. Построение тетрарного дерева
6.4.2. Восстановление изображения по тетрарному дереву
6.4.3. Сжатие изображений с помощью тетрарных деревьев
6.5. Бинарные деревья изображений
6.6. Алгоритмы расщепления — слияния
6.7. Построчное кодирование и граф смежности строк
6.8. Кодирование областей и граф смежности областей
6.9. Иконографические представления
6.10. Структуры данных для воспроизведения визуальных отображений
6.11. Библиографические замечания
6.12. Задачи
Приложение 6.а. Элементарные сведения о графах

Глава 7 Двухуровневые изображения
7.1. Введение
7.2. Дискретизация и топология
7.3. Элементы геометрии дискретной плоскости
7.4. Теорема отсчетов для изображений класса 2
7.5. Построение контура
7.5.1. Построение одиночного контура
7.5.2. Обход всех контуров области
7.6. Кривые и линии на дискретной сетке
7.6.1. Случай множества пикселов, не являющегося кривой
7.6.2. Случай множества пикселов, являющегося кривой
7.7. Кратные пикселы
7.8. Введение в задачи анализа формы объектов

7.9.Библиографические замечания
7.10.Задачи

Глава 8 Заполнение контура
8.1. Введение
8.2. Заполнение области между сторонами многоугольника
8.3. Заполнение контура по критерию четности
8.3.1. Доказательство корректности алгоритма 8.3
8.3.2. Реализация алгоритма заполнения области по критерию четности
8.4. Заполнение контура по критерию связности
8.4.1. Рекурсивная процедура заполнения по критерию связности
8.4.2. Нерекурсивная процедура заполнения по критерию связности
8.4.3. Процедуры, используемые при заполнении по критерию связности
8.4.4. Описание основного алгоритма
8.5. Сравнение и комбинации алгоритмов
8.6. Библиографические замечания
8.7. Задачи

Глава 9 Алгоритмы прореживания
9.1. Введение
9.2. Классические алгоритмы прореживания
9.3. Асинхронные алгоритмы прореживания
9.4. Реализация асинхронного алгоритма прореживания
9.5. Быстрый алгоритм прореживания
9.6. Структурный анализ формы объектов изображения
9.7. Преобразование двухуровневых изображений в штриховые
9.8. Библиографические замечания
9.9. Задачи

Глава 10 Построение по точкам и воспроизведение кривых
10.1. Введение

10.2. Интерполирование с помощью многочленов
10.3. Многочлены Безье
10.4. Определение многочлена Безье
10.5. Некоторые свойства многочленов Безье
10.6. Дуги окружностей
10.7. Воспроизведение прямых линий и кривых
10.7.1. Воспроизведение кривых на основе использования дифференциальных уравнений
10.7.2. Влияние ошибок округления на воспроизведение кривых
10.8. Программа-редактор точек
10.8.1. Структура данных для программы-редактора точек
10.8.2. Ввод и вывод при использовании программы-редактора точек
10.9. Библиографические замечания
10.10. Задачи

Глава 11 Построение кривых по точкам с помощью сплайнов
11.1. Введение
11.2. Основные определения
11.3. B-сплайны
11.4. Вычислительные аспекты использования в-сплайнов
11.5. Интерполяционные b-сплайны
11.6. Применение b-сплайнов в машинной графике
11.7. Описание формы объектов и b-сплайны
11.8. Библиографические замечания
11.9. Задачи

Глава 12 Аппроксимация кривых
12.1. Введение
12.2. Аппроксимация по критерию суммарной квадратической ошибки
12.3. Аппроксимация с помощью B-сплайнов
12.4. Аппроксимация с помощью сплайнов с переменными точками склеивания

12.5. Аппроксимации, основанные на использовании многоугольников
12.5.1. Субоптимальный алгоритм построения линий по точкам
12.5.2. Простой алгоритм построения многоугольника по точкам
12.5.3. Свойства алгоритма 12.2
12.6. Применение методов аппроксимации кривых в машинной графике
12.6.1. Обработка групп точек с помощью программы-редактора точек
12.6.2. Построение простых аппроксимирующих кривых
12.7. Библиографические замечания
12.8. Задачи

Глава 13 Приближение и воспроизведение поверхностей
13.1. Введение
13.2. Некоторые простейшие свойства поверхностей
13.3. Особые точки поверхности
13.4. Линейные и билинейные интерполяционные участки поверхности, ограниченные замкнутыми кривыми
13.5. Плазовые поверхности
13.6. Поверхности Кунса
13.7. Построение поверхностей по ориентирам
13.7.1. Поверхности Безье
13.7.2. Поверхности, построенные с помощью b-сплайнов
13.8. Выбор разбиения поверхности
13.9. Воспроизведение поверхностей и штриховка
13.10. Библиографические замечания
13.11. Задачи

Глава 14 Математические аспекты двухмерной машинной графики
14.1. Введение
14.2. Двухмерные преобразования

14.3. Однородные координаты
14.3.1. Уравнение прямой, задаваемой двумя точками
14.3.2. Координаты точки, определяемой пересечением двух прямых
14.3.3. Двойственность
14.4. Задачи, связанные с отрезками прямых
14.4.1. Положение точки относительно прямой
14.4.2. Пересечение отрезков прямых
14.4.3. Положение точки относительно прямоугольника
14.4.4 Тень отрезка
14.5. Библиографические замечания
14.6. Задачи

Глава 15 Разрезание многоугольников
15.1. Введение
15.2. Разрезание произвольного отрезка прямой произвольным выпуклым многоугольником
15.3. Разрезание произвольного отрезка прямой произвольным правильным прямоугольником
15.4. Разрезание произвольного многоугольника произвольной прямой
15.5. Пересечение двух многоугольников
15.6. Эффективное определение пересечения многоугольников
15.7. Библиографические замечания
15.8. Задачи

Глава 16 Математические аспекты трехмерной машинной графики
16.1. Введение
16.2. Однородные координаты
16.2.1. Положение произвольной точки относительно произвольной плоскости
16.2.2. Пересечение треугольников
16.3. Трехмерные преобразования

16.3.1. Некоторые сведения из векторной алгебры
16.3.2. Поворот относительно произвольной оси, проходящей через начало координат
16.4. Ортогональные проекции
16.5. Центральные проекции
16.6. Библиографические замечания
16.7. Задачи

Глава 17 Синтез трехмерных изображений в машинной графике
17.1. Введение
17.2. Задачи удаления невидимых линий и невидимых поверхностей
17.2.1. Тени поверхностей
17.2.2. Подходы к решению задачи разделения видимых и невидимых элементов сцены
17.3. Алгоритм разделения видимых и невидимых элементов сцены, основанный на использовании тетрарного дерева
17.4. Растровый алгоритм построчного разделения видимых и невидимых элементов сцены
17.5. Однородность элементов изображения по наблюдаемости
17.6. Описание нелинейных объектов
17.7. Придание воспроизводимому изображению естественного вида
17.8. Библиографические замечания
17.9. Задачи

Посвящается Полу,
Карен. и Харри

ПРЕДИСЛОВИЕ

Машинная графика и обработка изображений с помощью вычислительных машин приобрели популярность благодаря техническим достижениям последнего десятилетия. Значительный прогресс наблюдается также в распознавании «изобразительной информации». Совершенно очевидно, что интересы этих трех направлений исследований пересекаются Построение графических представлений имеет существенное значение для всех специалистов и пользователей, занимающихся обработкой изображений или их распознаванием, а решение многих задач машинной графики требует методологий, заимствованных из обработки изображений. Аналогичны и структуры данных, используемые в каждом из трех направлений Создается впечатление, что для этих направлений исследований имеется одна теоретическая основа, а именно *обработка изобразительной информации (изображений) с помощью вычислительных машин.*

Поскольку эти исследовательские направления — новые, то нелегко создать учебный курс или написать книгу, где были бы охвачены все их основные аспекты. В ряде учебников и монографий, посвященных машинной графике, внимание концентрируется на аппаратуре и частных методах, тогда как в учебниках и монографиях, посвященных обработке изображений, основной акцент делается на прикладные проблемы и классические методы обработки сигналов. Быстрое развитие технических средств приводит к тому, что публикуемый материал перестает соответствовать «текущему моменту». Например, появление оптических волокон делает менее существенным сжатие ширины полосы. Подобным образом уменьшение стоимости растровых графических устройств привело к резкой переоценке «ценностей» в области графики. В любом вводном курсе, необходимо сконцентрировать внимание на том материале, который может представлять интерес в течение длительного периода. Поскольку, судя по всему, математические средства, используемые в обработке изображений, отличаются наибольшим постоянством, именно им мы уделим основное внимание При этом мы не будем подробно останавливаться на тех ситуациях, когда аналитические средства должны быть дополнены знаниями, почерпнутыми из области эвристики или психологии зрительного восприятия. Анализу формы, скажем, посвящено лишь два раздела, несмотря на то, что мне эта тема чрезвычайно интересна и является основным предметом моих исследований С другой стороны, алгоритмы прореживания предназначены для достижения ясно

определенной цели и могут служить «рабочей лошадкой» при реализации многих схем распознавания. Я надеюсь, у читателя не возникнет впечатление, что прореживание является основой анализа формы. Подобным же образом, концентрация внимания на математических средствах не означает, что ими исчерпывается все необходимое при создании хорошей информационной системы для изображений. Существенны также хорошая осведомленность в области вычислительной техники, знание основ теории зрительного восприятия человека и некоторое понимание изобразительных искусств.

Большая часть материала, включенного в эту книгу, дважды прошла проверку при чтении мною лекционного курса в Принстонском университете в 1978 и 1980 гг. Аудитория состояла, главным образом, из студентов последнего и предпоследнего курсов, специализирующихся в области информатики[1]. Кроме того, мои лекции слушали несколько второкурсников и аспирантов, а также студентов технических, математических и естественных факультетов. Все они в целом имели хорошую подготовку в области вычислительных машин и являлись опытными программистами, поэтому я и не пытался включать в курс простейшие понятия информатики. Этот курс, помимо моих лекций, включал лабораторные занятия, лекции специально приглашенных специалистов и выездные занятия, что должно было позволить студентам получить представление о текущем состоянии дел в этой области В книге я постарался свести к минимуму материал, связанный с частными прикладными задачами и эвристиками.

Особая трудность, с которой сталкивается составитель подобного учебного курса, заключается в том, что обработка изображений связана с использованием обширного спектра математических и вычислительных средств. Математический анализ, элементы математической статистики, элементы теории графов, геометрия, методы обработки сигналов, структуры данных, методы исследования алгоритмов и программирование — все это необходимо. К сожалению, ранняя специализация не позволяет студентам получить необходимую подготовку. Это обстоятельство не только осложняет составление учебного курса, но и является серьезным препятствием для прогресса исследований в этом направлении. Например, проведение сегментации изображений требует определенной квалификации как в области обработки случайных сигналов, так и в. области структур данных и алгоритмов. Современные учебные программы редко позволяют студентам изучить и то, и другое.

Книга организована следующим образом. Вводная гл. 1 посвящена описанию видов изобразительных данных. Выделяются четыре вида таких данных, в общем, соответствующие тоновым изображениям (класс 1), двухуровневым, или бинарным изображениям.

[1] В данной книге мы будем пользоваться термином «информатика» для передачи английского *computer science,* что соответствует как организационным формам проведения этих исследований в СССР, так и сложившейся терминологической практике — *Прим. перев.*

(класс 2), кривым (класс 3) и точкам или многоугольникам (класс 4). Главы 2—4 посвящены тоновым изображениям и основное внимание в них уделяется преобразованиям и статистическим методам. В гл. 5 дается введение в методы восстановления изображений, используемые в машинной томографии. Необходимым условием понимания всех четырех глав, в которых речь идет только об изображениях класса 1, является вводный курс по обработке сигналов. Остальной материал книги не требует такой подготовки. В гл. 6 рассматриваются структуры для представления изобразительных данных, а в гл. 7—9 — обработка двухуровневых изображений и, в частности, такие проблемы, как построение и заполнение контура, прореживание. Все алгоритмы, представленные в этих трех главах, являются фактически алгоритмами обхода графа. Предполагается знакомство читателя с терминологией теории графов. Главы 10—13 посвящены построению кривых и поверхностей по точкам. Хотя формально изложенный в них материал базируется на простейших сведениях из анализа, все же некоторая математическая культура здесь не помешает. Главы 14—17 посвящены методам формирования графических отображений и их математическому фундаменту — линейной алгебре. В них рассматриваются в основном изображения класса 4.

Некоторые главы книги содержат больше материала, чем требуется для базового курса. Особенно это относится к гл. 8—11 и 15, в которые я включил дополнительно ряд важных алгоритмов, представляющих практический интерес для тех, кто занимается обработкой изображений и машинной графикой. В односеместровом курсе лучше ограничиться изложением одного алгоритма заполнения контура, прореживания и установления перекрытия многоугольников. Можно, в частности, воспользоваться простейшими алгоритмами: 8.4, 9.4 и 15.2. В подобном курсе, вероятно, можно рассматривать только многочлены Безье или только сплайны, или только сплайны с равномерным распределением узлов. (Минимальным введением в построение кривых по точкам можно считать разд. 10.1, 10.2, 11.1 — 11.3, 11.6 и 11.7, а также 10.8, взятые в указанном порядке.)

Если задачи учебного курса более ограничены, чем задачи нашей книги, то некоторыми главами можно вообще пренебречь. Так, учебный курс, посвященный машинной графике, может включать лишь гл. 1, 6—8, 10, 11 и 13—17, а учебный курс, посвященный обработке изображений — гл. 1—7 и 9; гл. 4, 9 и 12 особенно существенны для распознавания изображений.

Я писал эту книгу в течение трех с половиной лет (с конца 1977 г.) в трех местах: в Принстонском университете, Калифорнийском университете (Беркли) и, с июня 1980 г., в фирме Bell Telephone Laboratories, Марри-Хилл, шт. Нью-Джерси. Первый вариант рукописи был закончен в начале 1980 г. и использовался в качестве учебника по курсу, читавшемуся в Принстоне в весеннем семестре. Зимой 1980—1981 гг. этот текст был существенно переработан.

Мне посчастливилось воспользоваться помощью многих людей, которые были столь добры, что нашли возможность внимательно прочесть несколько вариантов рукописи. Даг Маклрой сделал множество замечаний как по содержанию, так и по «организации» текста. Он, в частности, предложил более элегантные доказательства для ряда утверждений и теорем. Карелии М. Бьёрклунд, Лоринда Л. Черри, Крис ван Уик и Ли-дэ У внесли множество конструктивных предложений по книге в целом. Материал, относящийся к построению кривых по точкам, был улучшен в результате замечаний, сделанных Карлом де Бором. Лэрри Шепп внес замечания по разделам, посвященным методам восстановления изображений. Кроме того, Лэрри С. Дейвис, Кеннет Фасман, Стайн Гринейкер, Р. Хилберт, Кристофер Ларсон и Марри Лою помогли мне, внеся ряд конструктивных предложений и обнаружив некоторые ошибки. Приятной неожиданностью оказалось то, что Карелии Ормс, редактировавшая рукопись, замечала ошибки, сделанные мной не только в языке, но также в математических формулах.

Матрицы, приведенные на рис. 3.8, составлены П. Ч. Чжянем. Большая часть изображений, использованных в примерах гл. 1—5, предоставлена мне Джоном Ф. Джарвисом; он же дал мне возможность пользоваться аппаратурой для документирования данных, воспроизводимых на экране дисплея. Я выражаю благодарность доктору Стэнли С. Сиджелману, Кену Ноултону, Тернеру Уиттеду и Дейвиду М. Ваймеру за предоставление мне оригиналов рис. 5.1, 6.8, 17.1 и 17.2 соответственно.

При подготовке рукописи и окончательного варианта книги было использовано программное обеспечение, действующее в Принстонском университете и в фирме Bell Laboratories в рамках операционной системы UNIX[1]. Лоринда Л. Черри исправила ошибки, имевшиеся в тексте, с помощью программ *spell, diction* и *style*[2]. Окончательный вариант был подготовлен с помощью программ форматирования *tbl, eqn* и *troff*. Рисунки 10.8, 11.4, 11.9, 12.1, 12.7, 12.8 подготовлены Брайаном У. Керниганом с использованием языка *pic, а* рис. 15.1 и 17.7 — Крисом ван Уиком с использованием языка *ideal*. Рисунок 10.1 получен на фотонаборном устройстве с помощью универсальной стандартной программы графического вывода.

И, наконец, я хочу выразить благодарность моей жене Мэрион и детям Полу, Карен и Харри за терпение и понимание, проявленные ими в тот период, когда я был всецело захвачен написанием этой книги. Очень часто их помощь оказывалась и более непосредственной: Мэрион ввела в систему UNIX большую часть текста, написанного от руки, а Пол показал приемы штриховки, используемые художниками (см. разд. 13.10).

Тео Павлидис Марри-Хилл, шт. Нью-Джерси
30 сентября 1981 г.

[1] UNIX — товарный знак фирмы Bell Laboratories.

[2] Названия программ переводятся как «чтение по буквам», «выбор слов» и «стиль» соответственно. — *Прим. перев.*

Глава 1

ВВЕДЕНИЕ

1.1. МАШИННАЯ ГРАФИКА, ОБРАБОТКА ИЗОБРАЖЕНИЙ И РАСПОЗНАВАНИЕ ОБРАЗОВ

Обработка информации, представленной в виде изображений, на ЭВМ имеет множество разновидностей и массу практических приложений. Эту область обработки информации обычно принято разделять на три направления: *машинную графику, обработку изображений и распознавание изображении.*

Задача *машинной графики* — воспроизведение изображении в тех случаях, когда исходной является информация неизобразительной природы, это направление имеет широкий спектр приложений. Сложность программ, так же как и вычислительные затраты, необходимые для получения соответствующих визуальных отображений, существенно зависят от характера конкретной задачи[1] Примеры визуализации (в порядке усложнения) включают построение графиков функций или экспериментальных данных, вывод информации на экран в получающих все большее распространение машинных играх и синтез сцен, предназначенных для использования в пилотажных тренажерах. Следует обратить внимание на то, что графики являются статическими относительно времени, визуальные отображения, встречающиеся в машинных играх, изменяются во времени, а сцены, используемые в тренажерах, не только изменяются во времени, но должны также создавать иллюзию глубины. Машинная живопись и машинный синтез мультипликационных фильмов представляют собой две сферы приложения машинной графики, требующие не только технической квалификации, но и таланта Сейчас, вероятно, именно в этих сферах наблюдается наиболее быстрое развитие Термин *интерактивная машинная графика* относится к устройствам и системам, в которые пользователь вводит исходные данные, сформулированные в терминах порождаемого визуального отображения (например, требуется провести на экране прямую между двумя точками, указанными пользователем).

[1] Читатель должен различать сложность программы и характеристик вычислительного процесса, называемую обычно вычислительной сложностью, т. е. вычислительные затраты, необходимые для выполнения соответствующей программы. Более подробные сведения по этому поводу можно найти в разд. 1.12,

Обработка изображений связана с решением таких задач, в которых и входные, и выходные данные являются изображениями. Одним из примеров служат системы передачи изображений — их разработчики сталкиваются с проблемами устранения шума и сжатия данных. Снимки, полученные с передержкой или недодержкой, как и размытые снимки, могут быть улучшены с помощью методов повышения контраста. Иногда желательно воспользоваться более радикальными преобразованиями. Изображение, характеризующееся большим диапазоном яркостей, можно свести к изображению, на котором будут видны только два уровня яркости. Силуэты, полученные в результате подобного преобразования, можно затем превратить в фигуры, построенные из палочек. В других случаях может возникнуть даже задача построения нового изображения по набору других изображений, как это требуется, скажем, при синтезе изображений поперечных сечений тела человека по рентгенограммам, соответствующим боковым проекциям

Задачей *распознавания изображений* является применение методов, позволяющих либо получить некоторое описание изображения, поданного на вход системы, либо отнести это изображение к некоторому определенному классу Распознавание зрительных образов представляет собой, в некотором смысле, обратную задачу машинной графики. Процедура распознавания применяется к некоторому изображению и обеспечивает преобразование его в некоторое абстрактное описание: набор чисел, цепочку символов или граф. Последующая обработка такого описания позволяет отнести исходное изображение к одному из нескольких классов Типичным примером прикладной задачи такого рода служит автоматическая сортировка почты, в процессе которой соответствующее устройство изучает почтовый индекс, написанный на конверте, и идентифицирует цифры, его образующие. Автоматизация медицинской диагностики связана с обнаружением на рентгенограммах или иных изображения медицинских объектов отклонений от нормы определенного вида

На рис 1.1 сведены воедино отмеченные выше сходство и отличия трех направлений обработки информации, представленной в виде изображений И хотя каждое из этих направлений имеет, по меньшей мере, двадцатилетнюю историю развития, их сходство привлекло внимание лишь недавно Связь распознавания и обработки изображений была впервые осознана благодаря возможности осуществления таких преобразований изображения, в резуль

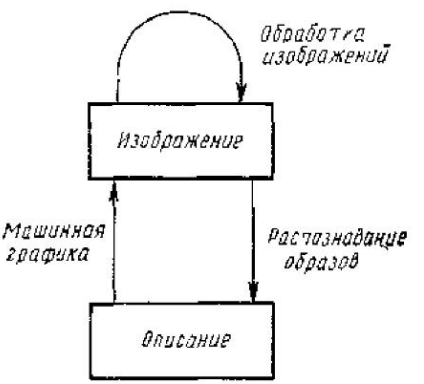


Рис 1 1 Схема, иллюстрирующая взаимосвязь машинной графики, обработки изображений и распознавания образов

тате которых упрощается задача его классификации. Осознание существования связей между этими двумя направлениями и машинной графикой произошло позднее. К одному из классов задач, несомненно являющимся предметом общего интереса этих трех направлений, относится получение внутреннего представления изображений в вычислительной машине (структуры данных, хранение и поиск, уплотнение и т. д.). В некоторых других задачах подобная общность интересов менее очевидна: так, при обработке изображений часто возникает задача *построения контура,* а в машинной графике столь же часто встречается задача *заполнения контура.* Поскольку эти операции взаимно обратны, то неудивительно, что определенные теоретические проблемы являются предметом общего интереса для этих двух направлений.

Приведем пример, в котором общность этих трех направлений проявляется в самой полной мере.

Пример 1.1. Существенное практическое значение имеет задача передачи изображений документов по каналам связи. Для обеспечения хорошего воспроизведения формы буквенно-цифрового символа необходимо преобразовать каждый символ в цифровую матрицу, содержащую, по меньшей мере, 20 строк и 15 столбцов (матрица покрывает и часть чистого пространства, окружающего символ). Для точного воспроизведения типографского шрифта (например, курсива, жирного шрифта и т. п.) требуется более высокая разрешающая способность. Указанная матрица может быть бинарной, и, следовательно, для передачи каждого символа необходимо затратить не менее 300 бит. Обработка изображений заключается в применении преобразований, отображающих в результате эту матрицу в объект, для передачи которого требуется меньше 300 бит, причем применение после приема обратного преобразования обеспечивает получение исходного изображения или, по крайней мере, его хорошего приближения. Использование известных преобразований такого рода может уменьшить число требуемых битов самое большее в шесть раз. Допустим, с другой стороны, что осуществлено распознавание и установлено название каждого символа, которое затем и передается на приемное устройство. Для описания такого названия обычно достаточно 8 бит и, таким образом, объем информации, подлежащей передаче, снижается почти в 40 раз. Приемное устройство, естественно, должно обеспечивать воспроизведение изображений символов, что составляет несложную задачу машинной графики. Отметим также, что для отображения символа можно было бы обойтись матрицей с меньшими размерами: обычно достаточно десяти строк и семи столбцов. Таким образом можно обеспечить «постоянное» сжатие данных. Задача становится более сложной, когда текст содержит иллюстрации. В этих случаях следует изменять систему кодирования в процессе обработки. Система, в которой реализован такой принцип, описана в работе *[1.5]*.

Пример 1.1 служит иллюстрацией задачи, при решении которой необходимо прибегать к обработке изображений, машинной графике и распознаванию. Подобные прикладные задачи становятся все более и более многочисленными по мере снижения стоимости использования вычислительных машин и периферийных устройств. Поскольку каждое из этих направлений достаточно обширно, невозможно охватить все три в одной книге. Вместо этого мы сосредоточим внимание на общих задачах и на тех проблемах, которые являются фундаментальными для обработки информации, представленной в виде изображений. Мы не будем затрагивать проблемы, связаные с аппаратурой, не только по причине стреми-

тельных изменений в области техники такого рода, но и потому, что мы делаем упор на алгоритмы и методы.

Одним аспектом нашей задачи прежде всегда пренебрегали — речь идет о своеобразии природы изображений, которое требует использования иных методов анализа по сравнению с применяемыми для обработки звуковых и электрических сигналов. Распространение методов обработки сигналов с одномерного на двухмерный случай — задача нетривиальная. Ее решение зависит и от физической интерпретации размерности. Так, сигнал, зависящий от времени и некоторой пространственной переменной, требует иной обработки, чем сигнал, зависящий от двух пространственных переменных. Математическое подобие форм сигналов может оказаться обманчивым.

1.2. ВИДЫ ДАННЫХ, ПРЕДСТАВЛЕННЫХ В ФОРМЕ ИЗОБРАЖЕНИЙ

Говоря об обработке изображений с помощью ЭВМ, целесообразно выделять четыре класса изображений. Эта классификация связана не столько с природой зрительного восприятия изображений, сколько с подходом к их представлению и обработке.

1.2.1. КЛАСС 1: ТОНОВЫЕ И ЦВЕТНЫЕ ИЗОБРАЖЕНИЯ

В класс 1 входят изображения, имеющие вид обычных телевизионных изображений. Они обеспечивают довольно точное воспроизведение «реальности» и представляются матрицами с целочисленными элементами (для обозначения последних обычно используют термины *элемент изображения, пиксел* и *пэл)*. В большинстве прикладных задач эти матрицы имеют очень большие размеры (512x512), являющиеся наиболее общепринятыми. В связи с этим представления изображений не всегда хранятся в памяти в виде обычных матриц — часто используются и более изощренные разновидности структур данных. Эта тема обсуждается в гл. 6. Цветные изображения могут представляться либо при помощи трех матриц (для красного, зеленого и синего цветов), либо с помощью одной матрицы таким образом, что отдельные биты каждого элемента представляют различные цвета. Поскольку человеческий глаз обычно не в состоянии различать уровни освещенности, отличающиеся друг от друга менее чем на 1%, то для представления цветного изображения достаточно затрачивать по одному байту на цвет на пиксел. Однако приемлемых результатов удается добиться, используя по 3 бита для передачи каждого из двух цветов и 2 бита для передачи третьего, так что в конечном счете для хранения изображения в памяти затрачивается лишь по одному байту на пиксел. Это мы более подробно обсудим в разд. 2.5. С математической точки зрения иногда удобнее рассматривать цветные изображения как матрицы трехмерных векторов.

1.2.2. КЛАСС 2: ДВУХУРОВНЕВЫЕ ИЛИ ПРЕДСТАВЛЯЕМЫЕ В НЕСКОЛЬКИХ «ЦВЕТАХ» ИЗОБРАЖЕНИЯ

Изображение книжной страницы служит типичным примером представителя класса 2—двухуровневого[1] (черно-белого) изображения. Подобные изображения можно представлять матрицами, затрачивая по одному биту на элемент, а также и в виде «карт», так как на этих изображениях имеются хорошо различимые области одного цвета. Именно поэтому мы объединяем в один класс изображения, представляемые в нескольких «цветах», и двухуровневые изображения, несмотря на то, что представление с затратой одного бита на элемент матрицы удачно лишь для двухуровневых изображений. Одна из проблем, возникающих в связи с использованием одного бита для представления каждого пиксела, заключается в отсутствии стандартного для различных типов ЭВМ и устройств визуального отображения способа объединения битов в байт и байтов в слово. Так, крайний слева пиксел может представляться как наименее, так и наиболее значимыми битами *байта.* Следовательно, пользователям всегда приходится думать о выборе способа представления, наиболее подходящего для применяемых ими устройств. Ниже опишем, каким образом карты могут быть представлены в ЭВМ Отметим, что различие между цветными и представляемыми Б нескольких цветах изображениями неопределенно и приобретает смысл лишь применительно к способу, используемому для представления изображения.

1.2.3. КЛАСС 3: НЕПРЕРЫВНЫЕ КРИВЫЕ И ЛИНИИ

Примерами изображений класса 3 служат контуры областей, сигналы и диаграммы (графики). Соответствующие данные являются *последовательностями точек,* допускающих представление через их координаты *x, y*. Однако этот метод представления довольно неэффективен, причем то же самое относится к представлению, основанному на использовании разностей значений координат • *x* и • *y* у соседних точек. Более эффективным является представление с помощью *цепных кодов,* при использовании которых вектору, соединяющему две соседние точки, ставится в соответствие один символ, принадлежащий некоторому конечному множеству. На рис. 1 2 показан обычный цепной код, использующий восемь направлений. Если точки расположены достаточно близко друг к другу, то ошибка, вносимая квантованием, может оказаться приемлемой. Если в качестве кодируемых точек выбираются смежные элементы матрицы изображения (вдоль строк, столбцов или диагоналей), то представление, основанное на использовании разностей индексов, указывающих положение элементов матрицы, требует затраты четырех

[1] Мы предпочитаем пользоваться термином «двухуровневое», а не «бинарное» изображение, хотя в литературе встречаются и тот, и другой. Слово «бинарный» вызывает некоторые ассоциации с ЭВМ и цифровыми устройствами и кое-кого удивляет его употребление применительно к аналоговым сигналам.

битов на точку (для представления значений —1, 0 и +1, принимаемых каждой координатой), тогда как представление с помощью цепного кода требует затраты лишь трех битов на точку.

Еще более эффективный способ представления заключается в применении *дифференциального цепного кода,* предусматривающего кодирование каждой точки разностью двух последовательных абсолютных кодов. В этом случае все еще сохраняется восемь значений (О, ±1, ±2, ±3 и 4), однако вероятности их появления неодинаковы. Можно ожидать, что при кодировании гладких кривых значения 0 и ±1 будут появляться много чаще, чем все остальные, а значение 4 — чрезвычайно редко. Поэтому для представления различных направлений можно воспользоваться каким-либо из *кодов с переменной длиной.* Приведем один из возможных вариантов кодирования словами с переменной длиной при использовании дифференциального цепного кода.

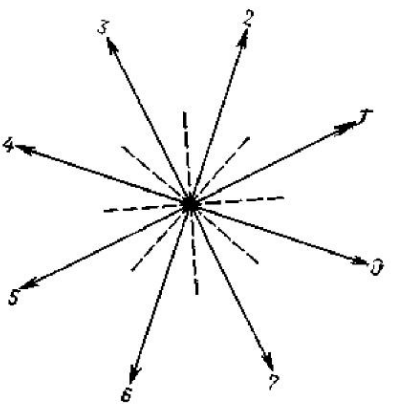


Рис. 1 2. Задание стандартною цепного кода; штриховые линии — направления, которые ставятся в соответствие некоторому определенному элементу кода

Направление	Код	Направление	
0	0	—2	01111
+1	01	+3	011111
—1	011	—3	0111111
+2	0111	4	01111111

При таком способе кодирования обычно затраты в среднем не превышают двух битов на точку. Проверка, проведенная автором па различных данных, показала, что для буквенно-цифровых символов средние затраты составляют 1,8—1,9 бит, а для очертаний таких объектов, как бокал на ножке или болт, эти затраты составляют в среднем 1,5—1,9 бит. Для небольшой окружности, являющейся неблагоприятным случаем, требуемые затраты равняются 2,3 бит/пиксел. Желательность применения цепного кода при обработке кривых зависит от конкретной задачи, однако заметим, что переход к координатам *x, y* и обратно не вызывает затруднений (см. задачу 1 4).

1.2.4. КЛАСС 4: ТОЧКИ ИЛИ МНОГОУГОЛЬНИКИ

Изображения класса 4 состоят из множеств отдельных точек, отстоящих друг от друга столь далеко, что для их представления цепным кодом пользоваться нельзя. Вместо него следует применять матрицу, содержащую их координаты *x, y.* Соответствующая аппаратура отображения позволяет соединять точки прямыми или простыми кривыми. Различие между классами 4 и 3, как и между предыдущими классами, довольно неопределенно и приобретает

смысл лишь при обращении к способу, используемому для представления изображения. Для представления точек, далеко отстоящих друг от друга, можно применять и цепные коды, требующие затраты более одного кодового символа на точку, причем выбор одной из двух форм представления должен основываться на характере распределения вероятностей расстояний между точками. Пусть l—среднее, а L — максимальное расстояния между точками. При этом кодирование с помощью цепного кода с переменной длиной кодового слова потребует затраты около $2l$ бит на точку изображения, а кодирование через приращения координат $\bullet x$ и $\bullet y$ — $2\log_2(L)$ бит на точку изображения. Если изменения расстояния между точками малы, то расстояние L не будет много больше l и, следовательно, описание изображения через разности координат окажется более эффективным.

В прикладных задачах машинной графики чаще всего используются изображения именно этого типа. Несмотря на то, что визуальное отображение может относиться к классу 2 или даже к классу 1, его внутреннее представление принадлежит классу 4. В особенности это относится к тем случаям, когда мы имеем дело с описаниями трехмерных объектов, спроектированных на плоскость экрана дисплея для получения воспроизводимого изображения. Во многих прикладных задачах используется одна из следующих форм представления.

а. Аппроксимация поверхности многогранниками; грани — обычно треугольники; после проектирования изображение состоит из многоугольников.

б. Криволинейная аппроксимация поверхности; на поверхности тела вычерчивается ряд кривых, описания которых затем используются для получения проекций, воспроизводимых в виде изображений класса 3.

в. Аппроксимация участками поверхности высшего порядка; этот способ аналогичен способу «а», за исключением того, что в качестве элементов, образующих поверхность объекта, используются не плоские многоугольники, а участки поверхности высшего порядка.

Во всех случаях положение объекта определяется некоторым небольшим числом точек и потому изображения класса 4 действительно представляют наибольший интерес для машинной графики.

1.3. ВВОД ИЗОБРАЖЕНИЙ

Изображение, представленное в аналоговой форме, необходимо преобразовать в некоторую числовую матрицу, прежде чем приступать к его обработке на ЭВМ. Процесс такого преобразования называется *дискретизацией* и состоит из двух процессов: *выборки* и *квантования.* Первый из них заключается в выборе на поле наблюдения некоторого множества точек. В каждой выбранной точке измеряются характеристики изображения, которые используются на

всех последующих этапах обработки изображения. Поскольку все ЭВМ располагают ограниченной памятью, результаты полученных измерений необходимо описать конечным числом символов — соответствующая процедура называется процессом квантования. Характеризуя плотность размещения выборочных точек, часто говорят о *пространственном разрешении,* а характеризуя точность представления результатов указанных измерений — о *тоновом* (или *цветовом) разрешении.* Выбор необходимых значений разрешения рассматривается в гл. 2, здесь же мы ограничимся кратким обзором применяемой аппаратуры и основных программных средств. Во многих устройствах дискретизации изображений используются телевизионные камеры, так как именно они обеспечивают преобразование светового сигнала в электрический. К последнему затем можно применять процедуры выборки и квантования, используя для этого *аналого-цифровой преобразователь* (АЦП). Единственная, возникающая при этом проблема — это колоссальная скорость воспроизведения данных на выходе стандартной телевизионной камеры. Системы телевизионного вещания передают 30 кадров в секунду, причем каждый кадр содержит около 500 строк растра. Если мы хотим обеспечить аналогичное разрешение по вертикали и горизонтали, то нам придется иметь дело с частотой выборки порядка 30X500x500 отсчет/с. Поскольку существуют аналого-цифровые преобразователи производительностью 10 млн. отсчет/с, то эта проблема имеет не столько технический, сколько финансовый характер. Кроме того, огромный поток данных может «переполнить» вычислительную систему, осуществляющую обработку изображений. Если вести обработку в реальном масштабе времени не обязательно, то можно перестроить телекамеру так, что она будет работать с меньшей скоростью, либо воспользоваться каким-либо устройством, осуществляющим сжатие полосы частот. Простейшим способом реализации такого сжатия является выборка по вертикали: переход к следующей вертикали осуществляется лишь после полного прохода по растру (или еще позже). Положение такой вертикали определяется вычислительной машиной, которая задает новое положение вертикали после того, как содержимое буферного запоминающего устройства полностью передано в постоянное запоминающее устройство. Этот прием позволяет уменьшить скорость поступления данных приблизительно в 500 раз.

В специализированных устройствах дискретизации изображений просмотр изображения в соответствующем порядке осуществляется при помощи прецизионного управления световыми пучками. В *барабанных сканирующих устройствах* изображение закрепляется на вращающемся барабане, а световой пучок перемещается параллельно оси барабана. Устройства дискретизации этого типа обычно работают медленнее тех, в которых используются телевизионные камеры, однако они обеспечивают получение результатов много лучшего качества. Устройства обоих типов воспроизводят изображения класса 1, хотя иногда применение грубого квантова

ния позволяет получать изображения класса 2. Другой тип устройств дискретизации изображений — *развертывающие устройства с бегущим лучом,* обеспечивающие построчный просмотр изображения. Кроме того, их можно запрограммировать так, чтобы они отслеживали границу между двумя областями изображения, отличающимися яркостью или цветом, и воспроизводили на выходе изображения класса 3. И наконец, *планшетные устройства дискретизации изображений* позволяют «проходить» по изображению линии и получать в результате дискретизированную последовательность координат пройденных точек линии. Этот способ дискретизации заключается в том, что исходное изображение налагается на намагниченную поверхность, а соответствующие точки указываются при помощи электрического пера или курсора. Изображение, получаемое в результате дискретизации, также относится к классу 3.

В некоторых прикладных задачах машинной графики необходимо осуществлять дискретизацию собственно объектов. Для этого обычно используются специализированные устройства, снабженные щупом, который в процессе дискретизации скользит по поверхности объекта; одновременно осуществляется выборка значений его координат. Другой метод дискретизации объекта предусматривает использование двух проекций объекта, которые одновременно обрабатываются на планшетном устройстве дискретизации изображений. Впоследствии описание трехмерного объекта можно восстановить по двум изображениям на плоскости, поскольку всякая точка некоторого пространства полностью определяется своими проекциями на две плоскости. Автоматизация выполнения таких процедур открывает многообещающие перспективы для взаимодействия распознавания образов, обработки изображений и машинной графики. Телевизионная камера позволяет получать различные виды объекта, а дальнейшая обработка этих изображений может привести к построению описания объекта. Этот процесс можно упростить, если вместо оптических изображений применить *данные о расстояниях* (до объекта). Для получения последних на объект направляется лазерный, радио- или ультразвуковой луч и отраженные сигналы, зарегистрированные в различных точках, используются для формирования изображения. Можно воспользоваться также методами восстановления изображений, рассмотренными в гл. 5.

Программное обеспечение (ПО), применяемое при дискретизации изображений, сравнительно несложно и не особенно отличается от ПО, применяемого при дискретизации данных любого другого вида. Обычно команда *считывания (управления внешним устройством, буферизации)* обеспечивает передачу данных от устройства дискретизации в буферный массив. В программе дискретизации по этой команде организован цикл, и выборка прекращается либо по специальному сигналу прерывания, либо после того, как получено заданное число отсчетов (см. задачу 1.1).

С другой стороны, часто оказывается удобным встраивать систему

ввода в *программу-редактор,* которая, как известно, позволяет пользователю изменять введенные данные. Например может возникнуть необходимость выделения подызображения вводимого изображения или изменения положения какой-то точки дискретизованной кривой. Для использования редактора необходимо иметь интерактивную графическую систему, причем сложность редакторов меняется довольно существенно в зависимости от того, какие возможности он должен предоставлять пользователю. Хотя простую программу-редактор можно написать за пару дней, для разработки более сложных программных комплексов требуется значительное число человеко-лет. Различные аспекты программ-редакторов будут рассмотрены в соответствующих разделах, а впервые мы займемся этой темой в разд. 1.9.

1.4. УСТРОЙСТВА ВИЗУАЛЬНОГО ОТОБРАЖЕНИЯ

Если дискретизация изображений осуществляется сравнительно просто, то преобразование данных в изобразительную форму представляет собой процесс значительно более сложный и для его реализации имеется множество весьма разнообразных устройств. Причина сложности этого процесса связана с тем, что информация поступает на вход устройства не обязательно в том виде, в каком это устройство должно ее воспроизвести, и, следовательно, оно не только должно выполнять *цифро-аналоговое преобразование* (ЦАП), но и осуществлять определенный объем обработки информации. Прежде всего следует различать устройства, выдающие машинные документы — *копии изображения* и устройства, снабженные *электронно-лучевой трубкой* (ЭЛТ). Для получения отпечатка копии изображения обычно используется всего один проход, поэтому входные данные должны быть упорядочены таким образом, чтобы обеспечивалась выдача соответствующих команд в последовательности, необходимой для надлежащего перемещения регистрирующего устройства. При воспроизведении отображения на экране ЭЛТ не обязательно строго соблюдать порядок воспроизведения, однако в этом случае необходимо многократно повторять процесс воспроизведения, чтобы воспроизводимое изображение можно было рассматривать столько времени, сколько потребуется. В *фотонаборных устройствах* используется комбинация двух этих процессов: изображение воспроизводится на экране ЭЛТ, а затем фотографируется для получения копии изображения.

Современные устройства на ЭЛТ снабжены микропроцессором: и буферной памятью. Главная ЭВМ загружает информацию, представляющую изображение, в буферную память, а затем микропроцессор просматривает память и воспроизводит изображение с частотой, необходимой для обеспечения зрительного восприятия. Если устройство визуального отображения не связано с ЭВМ (работает автономно), то для записи в буферную память используются любые доступные средства ввода. Содержимое буферной памяти

представляет собой команды для микропроцессора. Простейшим видом команды является команда *write (x, y, z)*, направляющая электронный луч в точку *(x, y)* экрана для получения там светящегося пятна, яркость или цвет которого должны быть пропорциональны некоторой величине z Для получения монохроматического отображения необходимо использовать, по меньшей мере, три ЦАП по одному на переменную *x, y* и *z*. Минимально допустимое число ЦАП, необходимое для получения цветных отображений, равно пяти: два для координат и по одному для передачи каждого цвета (переменная z теперь представляется некоторым вектором) Для обозначения содержания буферной памяти часто применяется термин *файл воспроизведения визуального отображения*[1] Этот файл представляет собой программу, в общем случае имеющую следующий вид

начало:

команды вида *write (x, y, z)*.

если в процессе выполнения программы прерывания не произошло, возврат к началу,

конец;

Обычно микропроцессор подвергает содержимое буферной памяти ряду преобразований и поэтому главная ЭВМ (или пользователь, работающий на автономных системах) не должна «заботиться» о деталях процесса получения отображения и может использовать для создания файла воспроизведения визуального отображения команды «более высокого уровня» Устройства, предназначенные для получения копий изображений, также могут располагать определенной вычислительной мощностью, которая, однако, обычно намного меньше, чем у устройств на ЭЛТ

Различие между воспроизведением изображения на экране ЭЛТ и получением его копии в виде машинного документа, которое может быть важным для пользователя, не столь существенно с точки зрения процесса преобразования исходных данных или вида файла воспроизведения визуального отображения. Основным является различие между *векторными* и *растровыми графическими устройствами.* (Векторные графические устройства иногда называют *каллиграфическими.)* На самом деле, предварительная обработка, специфичная для устройств, выдающих копии изображений, частично связана с переходом от векторного к растровому представлению и рассматривать ее лучше именно в этом контексте. Теперь мы дадим обзор основных характеристик обоих типов представления графической информации.

[1] В отечественной литературе иногда используется термин «дисплейный файл», являющийся дословным переводом английского термина *display file,* что по нашему мнению очень неточно передает смысл соответствующего понятия, поскольку речь идет не о файле дисплея, а о массиве данных, содержащих воспроизводимое изображение. — *Прим перев*

1.5. ВЕКТОРНАЯ ГРАФИКА

Векторные графические устройства воспроизводят изображения класса 3 (или 4). Элементарные команды обычно имеют следующий вид:

p (x, y) — направить электронный луч в точку с координатами *x, y,*

s(z)—задать интенсивность пучка электронов, равной значению *z*

Чтобы получить изображение объекта, необходимо сформировать последовательность команд такого типа, определяющую форму этого объекта Используемое устройство может располагать командами высшего уровня, каждая из которых эквивалентна группе элементарных команд, обеспечивающей воспроизведение какого-то стандартного вида объекта Как правило, именно так устроены команды, предназначенные для воспроизведения буквенно цифровых символов. Пусть S_1, S_2, , S_n — последовательности, соответствующие каждому из *n* объектов, составляющих конкретное отображение. В таком случае программа, реализуемая процессором, управляющим процессом воспроизведения, имеет следующий вид:

начало.

$S_1 S_2$

. . .

S_n

если в процессе выполнения программы прерывания не произошло, возврат к началу, конец;

При управлении воспроизведением t-го объекта необходимо корректировать лишь соответствующую ему последовательность S, Такой случай возникает, например, при перемещении одного объекта перед другими. При этом не нужно трогать ни одну другую последовательность. Существует множество способов выполнения коррекции последовательностей — простейшим является учет прерываний, генерируемых главным процессором или устройством ввода

начало.

$S_1 S_2$

. . .

S_n

если произошло прерывание, пропустите следующую команду

возврат к началу ввод данных

возврат к началу конец;

После каждого выполнения цикла изображение на экране, как говорят, *обновляется. Скорость обновления* обратно пропорциональна длине цикла. Если время между обновлениями велико по сравнению с временем затухания флуоресценции материала экрана, то наблюдаются мерцающие или тусклые изображения. Следовательно, существует верхняя граница длины цикла (и, таким образом, сложности воспроизводимого отображения).

1.6, РАСТРОВАЯ ГРАФИКА

На растровых графических устройствах воспроизводятся изображения классов 1 и 2; эти устройства могут также оснащаться устройствами, позволяющими пользователю имитировать процедуры векторной графики. Мы будем употреблять термин *растровые графические* применительно к устройствам, предоставляющим пользователю такую возможность, а устройства, не позволяющие воспроизводить векторные отображения, будем называть *дисплеями изображений.* Основной особенностью растровых графических устройств является наличие памяти большого объема, в которой но одной ячейке отводится на каждую адресуемую позицию экрана. Значения всех пикселов изображения вычисляются и вводятся в память, а затем при получении отображения эти значения выбираются из памяти и воспроизводятся в соответствующем месте экрана. Для обозначения памяти используются термины *кадровая буферная память* или *память обновления изображения* (или составленные из них комбинации). Основная команда такого устройства имеет следующий вид:

read (I, x, y, z) — произвести выборку данных из ячейки *I* (ее содержимое определяет значение *z; x и y* определяются по адресу ячейки *I*); затем выполнить команду *write (x, y, z]* .

При этом основной цикл воспроизведения отображения имеет следующий вид[1]:

```
начало:
        Для всех ячеек памяти I do: begin
                read (I, x, y, z) write
                (x, y, z) end
        возврат к началу
конец;
```

[1] Программы и алгоритмы приводятся в данной книге, как указывает ниже автор, в алголоподобной форме. Поэтому при переводе мы, как это принято, сохраняем английское написание так называемых основных символов языков программирования и операторов, вводимых автором, давая в сносках перевод их значений в случае использования нестандартных команд и операторов. — *Прим. перев.*

Если объем памяти обновления изображения равен N, то контроллер дисплея всегда должен выполнять N пар команд. В большинстве растровых графических устройств для воспроизведения изображений используются видеоконтрольные устройства, и чтобы мелькание изображения не было слишком сильным, необходимо выполнять цикл, по меньшей мере, тридцать раз в секунду. Все устройства такого рода оснащаются несколькими процессорами: набор команд, обеспечивающих обновление изображения, реализуется с помощью специальной аппаратуры, а другой процессор обеспечивает ввод в память описаний, поступающих от главной ЭВМ или пользователя. Таким образом, имеем фиксированную длину цикла и качество визуального отображения не зависит от его сложности. Однако описания объектов имеют взаимные «пересечения». При работе на векторных графических устройствах для устранения некоторого объекта необходимо только исключить соответствующую последовательность команд из цикла воспроизведения отображения. Не так просто обстоит дело в случае растровых графических устройств. В частности, нельзя задавать яркость всех пикселов объекта равной нулю, поскольку это одновременно приведет к стиранию других объектов, пересекающихся с удаляемым. Решение этой проблемы достигается благодаря введению *разделения памяти* и использованию отдельного раздела памяти для воспроизведения каждого объекта.

Существуют два способа воплощения этой идеи. Первый предусматривает наличие в памяти нескольких копий экрана. В системе разделения памяти, приведенной на рис. 1.3, все разделы памяти $(P_1 — P_4)$ имеют в качестве адресатов позиции экрана с одними н теми же координатами. Таким образом, пользователь может поместить описание каждого из четырех объектов в свой раздел памяти. Второй способ предусматривает *разделение слова.* В каждой ячейке памяти определенное число битов отводится под описание некоторого объекта. Если содержимое ячейки однозначно определяет яркость и цвет определенной позиции экрана, то при таком способе разделения задается цвет и яркость каждого объекта. Во многих растровых графических устройствах применяется еще один прием, позволяющий преодолеть указанное затруднение. Содержимое ячейки памяти / используется для адресации таблицы, элементы которой определяют цвет и яркость объекта. Содержание таких *справочных таблиц характеристик изображения* задается пользователем и может изменяться при переходе от одного отображения к Другому. Таким образом, для представления всех объектов можно использовать один диапазон цветов и яркостей.

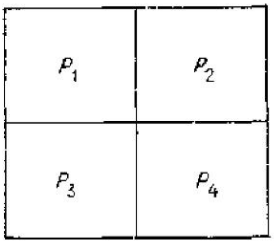


Рис. 1 3 Отделы памяти устройства визуального отображения; каждый отображается на один и тот же экран

Пример 1.2. Устройство работает с 12-разрядными словами, снабжено справочной таблицей и тремя ЦАП — по одному на каждый основной цвет. Четыре младшие бита элемента справочной таблицы отведены для синего цвета, следующие четыре — для зеленого и последние четыре — для красного. Простейшей формой задания такой справочной таблицы является *соответствие:* если значение, хранящееся в некоторой ячейке памяти, равно *n,* то выходное значение таблицы также равно я. Если в таком случае некоторая ячейка памяти содержит восьмеричное число 17, то соответствующее пятно на экране будет ярко-синим; восьмеричное число 360 приводит к появлению ярко-зеленого пятна, а восьмеричное число 7400 — ярко-красного и т.д. Если мы хотим воспроизвести сцену, состоящую из фона и двух перемещающихся объектов, то получение первого требует затраты трех битов на один цвет, а описания объектов — двух битое на один цвет. Для получения независимых отображений необходимо отвести определенные биты каждого слова на фон и объекты — в результате нам необходимо иметь 9+6+6=21 бит, т. е. больше, чем допускает память. Можно было бы воспроизводить эти изображения совмещая их, однако в этом случае при перемещении объектов пришлось бы хранить изображение фона в памяти обновления изображения. Можно было бы, с другой стороны, обеспечить получение независимых изображений объектов и фона, пожертвовав частью цветового диапазона: в каждой ячейке биты с нулевого по пятый отводятся на фон (по два на один цвет), биты с шестого по восьмой используются для представления первого объекта (по одному на цвет) и биты с девятого по одиннадцатый — для представления второго объекта. В этом случае восьмеричное число 1000 соответствует младшему биту второго объекта и, следовательно, пикселы, характеризуемые этим числом, можно представлять синим цветом. При этом соответствующее значение в справочной таблице может быть равно восьмеричному числу 10. Восьмеричное число 100 соответствует младшему биту первого объекта, отображение которого мы также задаем восьмеричным числом 10, и т. д. Схема разделения

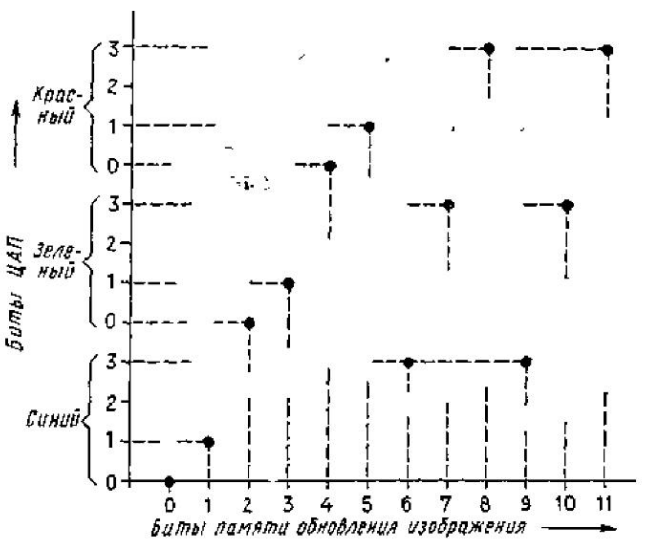


Рис. 1.4. Диаграмма, характеризующая привязку цвета и интенсивности к битам слова памяти обновления изображения для примера 1.2

слова в целом представлена на рис. 1.4. Обратите внимание, что мы не обязаны использовать лишь побитовое соответствие: справочная таблица может задавать, например, отображение восьмеричных чисел 100 и 1000 в восьмеричное

Таблица 1.1

Общепринятые элементарные команды воспроизведения графических объектов

	Название	Выполняемая функция
1	*setp (x, y)* (установка)	Поместить текущий воспроизводимый элемент изображения в позицию с координатами *x, y*
2	*vec (x, y)* (вектор)	Провести вектор от текущего воспроизводимого элемента изображения к элементу с координатами *x, y;* получаемое изображение относится к классам 3 или 4
3	*char (a)* (символ)	Воспроизвести буквенно-цифровой символ «а» в текущем элементе изображения
4	*win (v, y, X,* У) (окно)	Задать на дисплее прямоугольное окно, поместив его левый верхний угол в точку с координатами *x, y* и правый нижний угол в точку с координатами *X, Y*
5	*fore* (б) (передний план)	Установить яркость или цвет проводимых векторов (передний план) равными значению 6
б	*back (b)* (фон)	Установить яркость фона равной значению *b*
7	*wpic (buff)* (изображение в окне)	Воспроизвести содержимое буферной памяти *buff* в заданном окне. Каждый элемент буферной памяти порождает элемент изображения, и, следовательно, получаемое изображение относится к классу 1
8	*wrast (buff)* (изображение в окне черно-белое)	То же самое, что и в предыдущей команде, за исключением того, что каждый элемент изображения соответствует биту буферной памяти, и, следовательно, получаемое изображение относится к классу 2
9	*rpic (buff)* (передача изображения)	Передать изображение, воспроизведенное в окне, в буферную память *buff*
10	*erase* (стирание)	Приравнять яркость изображения в площади окна яркости фона
11	*readp (к, y)* (передача элемента изображения)	Зафиксировать координаты *x, y* элемента изображения, указанного пользователем
12	*enable (u)* (ключ)	Просмотреть слово *u* и после этого допускать изменение любыми другими командами только тех битов в словах памяти обновления изображения, которым в слове *u* соответствуют биты, равные единице
13	*color (u, v)* (раскраска)	С этого момента при появлении слова с двоичным набором *u* воспроизводить цвет, определяемый *v*

число 17 и т. п. Выбор такого решения предполагает, что имеется возможность избирательной адресации определенных битов каждого слова в памяти обновления изображения. В большинстве растровых графических устройств дело обстоит именно таким образом (см. команду 13 в табл. 1.1).

1.7. ОБЩЕПРИНЯТЫЕ ЭЛЕМЕНТАРНЫЕ КОМАНДЫ ВОСПРОИЗВЕДЕНИЯ ГРАФИЧЕСКИХ ОБЪЕКТОВ

Поскольку микропроцессор, управляющий процессом получения визуального отображения, может, как отмечалось, выполнять в некотором объеме обработку информации, пользователю не обязательно задавать файлы воспроизведения визуального отображения на самом элементарном уровне. Обычно в распоряжении пользователя имеется обширный набор команд. Так как книга посвящена не аппаратуре, а алгоритмам, будем называть такие доступные пользователю команды *элементарными* и в дальнейшем будем игнорировать существование команд низших уровней (типа тех, что упоминались в двух предыдущих разделах). В табл. 1.1 приведен перечень некоторых элементарных команд, используемых в графических устройствах обоих типов, и аббревиатуры, которые будут применяться для их обозначения. Мы используем термин *текущий воспроизводимый элемент изображения* для обозначения последнего элемента, направленного (адресованного) на отображение. Это понятие удобно применять для определения относительных расстояний.

Аргументами команд *setp* (1), *vec* (2) и *win* (4) могут быть либо *абсолютные,* либо *относительные адреса,* отсчитываемые относительно некоторых заданных значений. В качестве последних часто используются координаты текущего воспроизводимого элемента. Конкретная форма и название команды связаны с типом графического устройства, тем не менее команды, включенные в приведенный перечень, являются типичными Во всех векторных графических устройствах предусмотрены команды *setp* (1) и *vec* (2), а во многих и команды *char* (3) и *fore* (5). Эти команды типичны и для растровых графических устройств. Команда *readp* (11) используется в интерактивных графических устройствах. Она позволяет пользователю задавать на экране дисплея точку (позицию) с помощью светового пера или светового указателя, для управления которым применяются координатная ручка управления, координатный шар управления или иные устройства подобного типа[1]. Две последние команды удобно использовать для задания разделения слов в памяти растровых графических устройств и для задания

[1] Поскольку мы решили не обсуждать подробно аппаратуру, то устройства такого типа нами не рассматривались. Сейчас в продаже имеется множество разновидностей подобных устройств, и любая система обработки графической информации оснащена одним или несколькими из них. Не исключено, что появление сенсорных экранов сделает большинство таких устройств ненужными.

соответствия цветов и яркостей конкретным видам двоичных наборов (машинных слов) (вспомните пример 1.2).

Современные графические устройства часто обеспечивают пользователю дополнительные возможности работы с изображениями, например, такие, как поворот изображения, работа с парами изображений и т. д. Строго говоря, это уже не команды воспроизведения отображения, поскольку они включают операции, осуществляемые в памяти. Хотя в настоящее время на практике предусматривается оснащение дисплеев все большим и большим числом операций, мы будем исходить из того, что в нашем распоряжении имеются лишь перечисленные выше элементарные операции, а операции, связанные с реализацией нетривиальных алгоритмов, должны формироваться пользователем. Поскольку эта книга предназначена в первую очередь для инженеров, а не для обычных пользователей, работающих с графическими устройствами, соответствующие методы будут излагаться подробно.

1.8. СРАВНЕНИЕ ВЕКТОРНЫХ И РАСТРОВЫХ ГРАФИЧЕСКИХ УСТРОЙСТВ

Главным достоинством векторных графических устройств является разделение описания объекта и экрана. Таким образом, разрешающая способность такой системы (т. е. минимальное расстояние между двумя точками экрана) определяется электронной частью аппаратуры, используемой для записи на экране. Так, общепринятой считается экранная матрица размерами 4096X4096[1]. Можно также использовать движение луча для проведения прямых между двумя точками в *аналоговом,* а не цифровом режиме.

С другой стороны, в растровых графических устройствах предусматривается взаимно-однозначное соответствие между пикселами экрана и ячейками памяти обновления изображения. В результате любое повышение разрешающей способности требует увеличения объема памяти — в настоящее время максимальные размеры используемых экранных матриц составляют 1024X1024. Прямые могут воспроизводиться только в цифровом режиме, и это обстоятельство в сочетании с ограниченностью разрешающей способности приводит к изображению «ступенчатого» вида.

Другой недостаток растровых графических устройств заключается в том, что, как бы ни была разделена память — по адресам или по структуре машинных слов, число выделяемых разделов памяти невелико и, следовательно, невелико число объектов, которые можно обрабатывать независимо.

Растровые графические устройства позволяют без затруднений воспроизводить изображения классов 1 и 2. Для этого необходимо лишь ввести соответствующие значения в память обновления изображений

[1] Фотонаборное устройство, использованное при выпуске нашей книги, относится к типу векторных графических устройств; его адресный диапазон составляет приблизительно 0—8000.

Изображения класса 1 невозможно воспроизводить с помощью векторной графической системы, не превратив ее в сущности в растровую графическую систему. Ценой определенных усилий можно воспроизвести изображения класса 2: необходимо покрыть каждую область заданного цвета близко расположенными параллельными линиями.

1.9. ПРОГРАММА-РЕДАКТОР ИЗОБРАЖЕНИЙ

Обсуждая в последних двух разделах ввод и вывод изображений, мы касались лишь простейших аспектов этих операций. При решении большинства практических задач желательно располагать чем-то большим, чем система, которая просто дискретизирует изображения. Целесообразно, как отмечалось в разд. 1.3, скомпоновать все операции ввода, создав некую *программу-редактор изображений.* Мы используем этот термин для обозначения программы, которую можно применять для ввода и модификации изображений точно таким же образом, как *программа-редактор текста* применяется для ввода и модификации текста[1]. Наличие такой программы-редактора обеспечивает также удобные средства для реализации и проверки различных алгоритмов обработки изображений. Результаты работ такого рода можно включать в редактор с тем, чтобы расширить набор возможностей, предоставляемых им пользователю[2]. С практической точки зрения, вероятно, необходимо иметь два редактора: *программу-редактор изображений* для работы с объектами классов 1 и 2 и *программу-редактор точек* для работы с объектами классов 3 и 4. Редакторы точек мы рассмотрим в конце гл. 10, а здесь займемся редакторами изображений. Существенными компонентами систем такого рода являются устройство дискретизации изображений, дисплей и устройство, указывающее пикселы, т. е. реализующее команду *readp (x, y)* из табл. 1.1. Выбор подызображений, вероятно, лучше осуществлять цифровыми, а не аналоговыми средствами. Таким образом, устройство дискретизации может передавать свои выходные данные непосредственно в память обновления изображений дисплея. Затем пользователь задает на экране некоторое окно и применяет ко-

[1] Употребления термина *графический редактор* лучше избегать, поскольку он может иметь два значения: а) программа для редактирования вводимых изображений; б) редактор текста, предназначенный для графического дисплея.

[2] Именно поэтому разработка такой программы-редактора является идеальной задачей для курсовой работы по вычислительному практикуму. Практические занятия играют очень важную роль в изучении машинной графики ч обработки изображений, причем существенных результатов можно добиться простыми средствами. Так, в гл. 5 рассмотрена довольно сложная задача, при решении которой для вывода результатов необходимо использовать только построчно-печатающее устройство. Поскольку значительная часть работы связана с процедурами ввода и вывода, имеет смысл объединять такие задачи в виде одного редактора.

команду *rpic (buff)* (передать изображение в буферную память), чтобы сохранить соответствующее подызображение.

Базовая структура любой программы-редактора предусматривает наличие некоторого цикла, выполняемого многократно. На первом шаге цикла обычно вырабатывается побуждающее сообщение, в ответ на которое пользователь должен ввести некоторую последовательность символов, описывающих подлежащую выполнению операцию. После этого программа просматривает введенную последовательность[1] и вызывает соответствующую подпрограмму. Иногда для указания необходимой команды удобно пользоваться *меню* световых кнопок (команд) графического дисплея. В этом случае пользователь не вводит команду с клавиатуры, а просто указывает некоторую область на экране. На рис. 1.5 показано, как

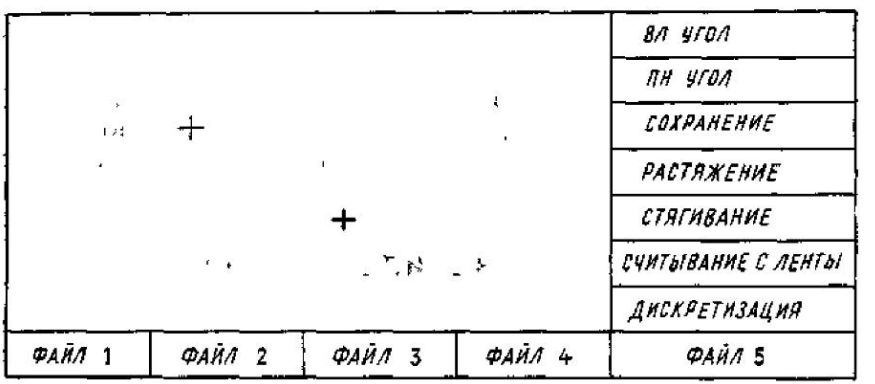


Рис. 1.5. Отображение на экране дисплея при использовании простой программы-редактора изображений:
+ световые указатели, позволяющие пользователю проводить адресацию точек экрана, просто указывая их

может выглядеть отображение на экране дисплея при использовании простой программы-редактора. Левая часть экрана оставлена для воспроизводимого изображения, а в правой помещается перечень команд меню световых кнопок. Мы предполагаем, что пользователь имеет возможность перемещать по экрану два указателя и может сообщить системе, что следует зафиксировать координаты указателя (команда *readp (x, y)*. В нижней части экрана размещается перечень названий файлов.

Типичным является следующий режим работы. Пользователь переводит первый указатель в ту часть экрана, где воспроизведено меню световых кнопок, и задает системе исполнение команды *readp (x, y)*. После этого программа сопоставляет значения

[1] Следовательно, основной цикл состоит из последовательности операторов "if ... then" или switch в языке Си, case в Паскале, вычисляемых операторов GOTO в Фортране и т. д.).

координат *x, y* со значениями перечня команд, чтобы выбрать заданную пользователем команду. Команды READ TAPE (СЧИТЫВАНИЕ С ЛЕНТЫ) и DIGITIZE (ДИСКРЕТИЗАЦИЯ) побуждают программу считывать с ленты файл или вводить данные с помощью устройства дискретизации. В обоих случаях изображение, получаемое в результате выполнения заданной операции, воспроизводится на экране. Если задаются операции ВЛ УГОЛ или НП УГОЛ, то программа применяет команду *readp (x, y)* к положению второго указателя, что позволяет определить значения координат верхнего левого или правого нижнего углов соответственно. Команда SAVE (СОХРАНЕНИЕ) побуждает программу зафиксировать координаты второго указателя, чтобы определить название файла для сохраняемого изображения. Последним является изображение, Ограниченное прямоугольником, который задается координатами своих верхнего левого и нижнего правого углов Команда SHRINK (СТЯГИВАНИЕ) обеспечивает замену изображения, заданного указанным прямоугольником, изображением (приблизительно) в одну четвертую исходного размера, а команда EXPAND (РАСТЯЖЕНИЕ) обеспечивает противоположный результат[1]. Способы реализации этих команд, а также некоторых других рассматриваются в гл. 2—4.

1.10. ПРЕОБРАЗОВАНИЯ ИЗОБРАЖЕНИЙ

Множество задач, приводимых в литературе, можно представить как преобразования между классами изображений, а некоторые задачи — как преобразования, осуществляемые в пределах классов изображений. В данном разделе будет дан краткий обзор таких преобразований.

Преобразование изображения класса 1 в изображение класса 2. Этот процесс называется *сегментацией* и обеспечивает выделение областей приблизительно одинаковых цвета и/или яркости. Часто термин «сегментация» используется для обозначения процесса поиска однородности в смысле некоторого более сложного свойства (типа текстуры). Мы будем заниматься лишь простыми схемами сегментации, которые представлены в гл. 4.

Преобразование изображения класса 2 в изображение класса 3. Одним из допустимых преобразований этого типа является *построение (отслеживание) контура, а другим — прореживание.* Первое обеспечивает отображение заданной области в некоторую замкнутую кривую (см. гл. 7), а второе (см. гл. 9) — отображение области в некоторый граф, называемый *остовом* области.

Преобразование изображения класса 3 в изображение класса 4. Процесс такого преобразования, называемый иногда *сегментацией*

[1] Можно было бы выбрать и другой масштаб — не обязательно одну четвертую; можно также предоставить пользователю выбор степени стягивания или растяжения изображения из некоторого заданного списка масштабных коэффициентов.

кривых, предназначен для отыскания *критических точек* контура. В случае многоугольников этими точками являются углы. Подобные преобразования обычно рассматриваются в литературе, посвященной распознаванию образов, и для их реализации может потребоваться применение сложных математических методов. В гл. 12 обсуждается несколько простых схем отыскания критических точек кривых.

Преобразование изображения класса 4 в изображение класса 3. Это преобразование включает процессы *интерполирования* (см. гл. 10 и 11), обеспечивающего проведение гладкой кривой через некоторое множество точек, и *аппроксимации* (см. гл. 12), обеспечивающей проведение гладкой кривой вблизи некоторого множества точек.

Преобразование изображения класса 3 в изображение класса 2. Если в качестве входной информации задается контур, то возникает задача *заполнения* контура, которая часто рассматривается как задача *штриховки.* В последнем случае яркость или цвет некоторой области не оставляются одинаковыми, а изменяются в соответствии с определенными правилами. Методы решения этой задачи приводятся в гл. 8. Если входной информацией служит остов, то для восстановления области следует использовать процедуру *расширения.*

Преобразование изображения класса 2 в изображение класса 1. Изображение, воспроизведенное на экране в нескольких цветах, часто оказывается ущербным в эстетическом отношении, так как глаз человека легко обнаруживает контуры (см. рис. 2.9 и 2.10). Некоторого сглаживания изображения можно добиться с помощью *фильтров нижних частот* или подмешивания *низкочастотного* шума. Эти методы рассмотрены в гл. 2 и 3.

Грубо говоря, преобразования, переводящие изображения из классов с меньшими номерами в классы с большими номерами, откосятся к сфере интересов распознавания образов, а преобразования, переводящие изображения из классов с большими номерами в классы с меньшими номерами — к сфере интересов машинной графики. При обработке изображений используются и те, и другие преобразования, а также преобразования, не выводящие изображение за пределы соответствующего класса. *Улучшение качества изображения* является, например, *внутриклассовым преобразованием,* а *сжатие изображения* часто оказывается преобразованием, переводящим его из класса 1 в класс 2.

Еще один важный класс задач возникает в связи с преобразованиями, связывающими двухмерные изображения и трехмерные объекты. Термин *проектирование* используется для обозначения операции, при помощи которой трехмерный объект преобразовывается в двухмерное изображение или, в частном случае, двухмерное поперечное сечение объекта — в одномерный массив. Для обозначения операции восстановления трехмерного объекта (либо его поперечного сечения) по его проекциям используется термин *обратное*

проектирование. Эти задачи играют важную роль, главным образом, в двух прикладных областях.

В аксиальной поперечной томографии поперечное сечение трехмерного объекта восстанавливается по набору рентгеновских проекций. Для обозначения процедур, обеспечивающих решение этой задачи, употребляется термин *алгоритмы восстановления*. Они рассматриваются в конце гл. 5.

В машинной графике часто возникает потребность воспроизвести некоторую проекцию трехмерного представления пространственного объекта. Совсем нетрудно выполнить необходимые геометрические преобразования и определить проекции всех точек этого объекта на плоскость отображения. Полученное изображение не будет, тем не менее, являться искомой проекцией, так как на нем представлены все точки объекта, несмотря на то, что в действительности часть из них заслонена другими точками и, следовательно, не может быть видна. Необходимо, таким образом, располагать алгоритмами удаления *невидимых линий* и *невидимых поверхностей*. Этим проблемам посвящены гл. 16 и 17. Как показывает следующий пример, эти две задачи иногда объединяются.

Пример 1.3. Пусть врачу требуется несколько различных проекций сердца пациента. Аксиальную поперечную томографию можно использовать для получения последовательности изображении поперечных сечений сердца, объединение которых дает затем описание сердца. Система машинной графики позволяет после этого воспроизвести изображение изучаемого органа в различных проекциях.

Преобразования, не выводящие изображение за пределы соответствующего класса, как правило, отличаются сравнительной простотой Вот несколько примеров таких преобразований.

Преобразования изображении, сохраняющие их в пределах классов 1 и 2. Фильтрация. Это преобразование включает повышение контраста, удаление высокочастотного шума и т. д.

Преобразования изображений, сохраняющие их в пределах классов 3 и 4. Изменение системы координат. Это преобразование включает повороты и переносы.

Преобразования изображений, сохраняющие их в пределах любого класса. Разложение в ряды. Наиболее распространенным в этой группе является преобразование Фурье. Преобразования этой группы часто используются для сжатия данных.

1.11. СИСТЕМА ЗАПИСИ АЛГОРИТМОВ

В нашей книге алгоритмы приводятся в алголоподобной записи с учетом следующих соглашений: ключевые слова печатаются жирным шрифтом, переменные — курсивом, а весь остальной текст алгоритма — прямым светлым шрифтом. В круглые скобки заключаются индексы массивов, а также аргументы функций и процедур. В фигурные скобки { } заключаются комментарии. Запись *f (*, I)* является обозначением всех элементов /-го столбца двухмерного массива f. Реализация приводимых алгоритмов на Алголе, ПЛ/1, Паскале и языках Си и Ратфор не должна вызывать затруднений. Несколько сложнее может оказаться их реализация на Фортране, тем не менее мы полагаем, что в 80-х годах

большинство людей владеет тем или иным языком программирования или знает, каким образом структурированные алгоритмы записываются на Фортране.

Проиллюстрируем используемую нами систему записи на примере алгоритма, обеспечивающего получение изображения класса 1 при помощи печати с наложением. Этот способ вывода изображений являлся хорошим подспорьем для исследователей в период, предшествовавший распространению растровых графических устройств; еще и сегодня он остается популярным хобби. Этот способ целесообразно использовать также в процессе обучения, поскольку построчно-печатающие устройства все еще являются более стандартным оборудованием, чем растровые графические дисплеи. Предполагается, что известна некоторая матрица С, определяющая кодировку печати с наложением. Все символы, содержащиеся Б *J*-м столбце этой матрицы, налагаются при печати для получения тона *j*. Хорошим вариантом матрицы *С* является матрица символов, используемых для воспроизведения тонов серой шкалы при помощи печати с наложением:

M M M MM MMM H H HH H X H X O Z WMNH0OS=I*+ + =:—.—'
W WW WW WWW≠≠≠* + +———-— = — —
≠≠≠≠≠≠≠ O O+—
O O O O O
0 0 ┬
+

В записи алгоритма буква М обозначает число строк матрицы *С*.

Алгоритм 1.1. Получение тоновых изображений при помощи печати с наложением.

1 For каждая строка f(*) изображения do: {Мы не заботимся в данном случае о том, каким образом определяется *f*, поскольку это зависит от конкретной задачи.} **Begin.**

2. For *i* от 1 до *M* do

Begin.

3 For *k* от 1 до *n* do: {*n* — число элементов в строке изображения.}

Begin.

4 Установка *p (k)* равным *C (i, f (k))*.

End

5. Печать массива *p* без протяжки бумаги на следующую строку.

End.

6 Протяжка бумаги на следующую строку.

End 7 Конец алгоритма.

Операторную конструкцию 1 можно реализовать либо посредством цикла **for/do,** если известно число строк, либо посредством Цикла **while,** в котором проверяется окончание файла. Операторная

конструкция 2 очевидна, а операторную конструкцию 3 можно реализовать посредством цикла for/do, или while, если конец строки отмечается нуль-символом. В большинстве языков операторная конструкция 4 будет иметь вид $p(l) = C(j, f(l))$, причем в Фортране для ее реализации может потребоваться два шага. И, наконец, реализация операторной конструкции 5 предполагает, что символы управления кареткой известны. В языке ПЛ/1 следует опускать команду SKIP, а в Фортране, при использовании форматированного вывода, первым символом должен быть «+». Приведенный алгоритм основан на предположении о равенстве числа столбцов матрицы C числу допустимых шкал полутонов. Если это допущение оказывается неверным, то необходимо предусмотреть соответствующие процедуры проверки индексов. Мы не будем включать в алгоритмы подобные процедуры, поскольку рассматриваем их просто в качестве приемов программирования, позволяющих получать хорошие программы (см. например, [1.3]). В данном случае может оказаться целесообразным воспользоваться несколько более сложным отображением g, связывающим уровни полутонов и столбцы матрицы C; в результате в операторную конструкцию шага 4 придется ввести запись $C(j, g(f(l)))$. Выбор такого отображения рассматривается в разд. 3.2. Более эффективным способом является расширение исходной матрицы C таким образом, чтобы ее размерность соответствовала числу допустимых уровней полутонов. Однако, описывая алгоритмы, мы таких проблем затрагивать не будем, поскольку они относятся к качеству и стилю программирования.

Если читатель захочет реализовать алгоритм 1.1, он может воспользоваться приведенной здесь таблицей значений матрицы C. Она была предложена Хендерсоном и Танимото [1.2]; тот, кто хочет попытаться реализовать данный алгоритм на ассемблере, найдет в их статье интересную для себя информацию.

1.12. НЕСКОЛЬКО ЗАМЕЧАНИЙ ОТНОСИТЕЛЬНО СЛОЖНОСТИ

Всякое обсуждение алгоритмов должно затрагивать проблемы вычислительной сложности и потому, излагая какой-либо алгоритм, мы всегда будем стараться давать некоторое представление о времени и пространстве, необходимых для его реализации. Я хотел бы в этой связи отметить одну ошибку, которую часто совершают многие студенты: речь идет о смешении вычислительной сложности и сложности программирования. Длина программы, реализующей алгоритм, в принципе, мало связана с быстротой его выполнения и даже с требованиями к памяти. Если какая-то связь и существует, то она скорее имеет противоположный характер. «Сложные» алгоритмы обычно быстрее «простых». Так, (нерекурсивная) программа для выполнения быстрого преобразования Фурье и длиннее, и логически сложнее, чем программа, реализующая формулу суммирования этого преобразования, одна-

ко выполняется она много быстрее. Подобные примеры являют нам и алгоритмы сортировки.

Часто представляется заманчивым использовать рекурсивное представление алгоритма, поскольку оно много короче нерекурсивного, а число операций в обоих случаях одинаково. При этом не следует забывать о высокой стоимости рекурсивных обращений, необходимости устройства для хранения содержимого регистров и т. д. Если число подобных обращений невелико по сравнению с числом прочих операций, то эти затраты вполне могут быть оправданы простотой получаемой в результате программы. В противном случае нерекурсивное представление позволяет получать более эффективные программы.

Если простота программирования и может завлечь студента, который должен закончить программирование к определенному сроку и предполагает прогнать программу, используя ограниченный набор данных, то при решении прикладных задач, когда программа обрабатывает большие объемы данных, такая практика пагубна.

1.13. БИБЛИОГРАФИЧЕСКИЕ ЗАМЕЧАНИЯ .

Устройство дисплеев, систем дискретизации изображений и тому подобной аппаратуры рассматривается во многих учебниках по обработке изображений и машинной графике; для первого чтения хорошо подходят учебники [1.1, 1.4]. Ситуация в этой области изменяется быстро и поэтому целесообразно тщательно следить за тем, что поставляют на рынок производители аппаратуры. Хорошим источником информации являются издания профессиональных обществ и труды конференций. В США специализированная группа по графике Ассоциации по вычислительной технике (The ACM Special Interest Group on Graphics — SIGGRAPH) публикует ежеквартальное издание *Computer Graphics* (Машинная графика) и ежегодно организовывает конференцию (обычно летом), в рамках которой устраивается выставка основного оборудования. Подробности, касающиеся этих конференций, можно найти в разделе «Календарь» журнала *Communications of the ACM*. В такие публикации Института инженеров по электронике и радиоэлектронике, как журналы *Spectrum, IEEE Computer Graphics* и *Computer,* всегда включаются разделы, содержащие рекламу и сообщения о новом оборудовании; весьма часто в этих разделах встречаются сведения об оборудовании для обработки изображений и машинной графики.

Кроме того, научные статьи, посвященные обработке изображений, печатаются в журналах *Computer Graphics and Image Processing* (сокращенно CG/P)[1], *IEEE Transactions on Pattern Ana-*

[1] С 1983 г, (ст 21) этот журнал выходит под названием *Computer Vision, Ciaphics and Image Processing* (сокращенно CVGIP). — *Прим перев.*

lysis and Machine Intelligence, а также в недавно объявленном журнале ACM *Transactions on Graphics* и др.[1]

1.14. ЗАДАЧИ

1.1. Напишите программу дискретизации для одного из устройств, входящего в комплект имеющейся у Вас системы.

1.2. *Упаковка битов.* Пусть в тоновом изображении присутствуют лишь два различных уровня серого тона, так что кодировать его можно как двухуровневое изображение. Допустим далее, что всем пикселам присвоены значения 0 или 1. Напишите программу запоминания этого изображения, предусматривающую затрату одного байта на восемь пикселов

1.3. *Распаковка битов.* Для воспроизведения предыдущего изображения необходимо затрачивать по одному байту на каждый пиксел. Напишите программу «распаковки».

1.4. *Цепной код.* Напишите программу определения координат x и y пиксела, для которого известно направление цепного кода связи с другим пикселом. Напишите также программу для обратной операции — определения цепного кода по координатам x и y.

1.5. Напишите простую программу-редактор, описанную в разд 1 9, но не включайте в нее команды РАСТЯЖЕНИЕ и СТЯГИВАНИЕ. Следует попытаться сделать программу в максимальной мере независимой от устройства отображения. Так, можно, ввести процедуру, идентифицирующую команду по координатам светового указателя. Вместо вызова процедуры для выполнения соответствующей операции процедура ввода должна передавать цепочку, содержащую название команды, в основную программу. Таким образом устраняется необходимость загружать основную программу-редактор подробностями, относящимися к графическим командам.

1 6. *Получение серого тона при помощи печати с наложением* Реализуйте алгоритм 1.1. При желании можно воспользоваться кодом, приведенным в разд. 1.11 для матрицы С.

Глава 2

ДИСКРЕТИЗАЦИЯ ТОНОВЫХ ИЗОБРАЖЕНИЙ

2.1. ВВЕДЕНИЕ

Если какое-то изображение необходимо обработать с помощью ЭВМ, то оно часто представляется матрицей или иной дискретной структурой данных. Однако изображение в первую очередь есть некоторый сигнал, передающий информацию некоторому наблюдателю, причем во многих прикладных задачах это условие является чрезвычайно существенным. Эта и следующая главы посвящены обсуждению таких задач, в частности применительно к тоновым изображениям (изображения класса 1). Первая задача состоит из преобразовании непрерывного изображения в дискретную форму и включает две процедуры: *выборку,* которая сводится к выбору

бору дискретной сетки для представления изображения, и *квантование,* заключающееся в отображении значений яркости и цвета в целые числа. В машинной графике возникают аналогичные задачи: в частности, выбор разрешения визуального отображения и числа уровней серого тона или цветов. Эти процедуры используются также при обработке одномерных данных и соответственно тщательно изучаются применительно к этому случаю, но при работе с двухмерными данными возникают новые задачи. Первая часть главы посвящается обзору методов преобразований, затем рассматривается выборка, для одномерного случая, после чего — выборка для случая изображений. В последнем разделе главы обсуждается квантование.

2.2. ПРЕОБРАЗОВАНИЯ ФУРЬЕ И ПРЕОБРАЗОВАНИЯ ДРУГИХ ВИДОВ

Преобразования Фурье являются стандартным инструментом обработки сигналов и определены для одномерных и двухмерных функций. Для непрерывного случая справедливы следующие две формулы: если $f(t)$ — функция, определенная на интервале $[0, \infty)$, то ее фурье-образ имеет вид

$$F(w) = \int_{t=0}^{\infty} f(t)\, e^{-iwt}\, dt \quad 0 \leqslant w < \infty. \tag{2.1}$$

Если $f(x, y)$ — функция двух переменных определенная на бесконечной плоскости, то ее образ при двухмерном

$$F(u, v) = \int_{-\infty}^{\infty} \int_{-\infty}^{\infty} f(x, y)\, e^{-i(xu+yv)}\, dxdy \quad 0 \leqslant u, v < \infty. \tag{2.2}$$

В качестве синонима понятия преобразования Фурье часто ис-

$$F(u) = \sum_{k=0}^{N-1} f(k) \exp\left(-i\,\frac{2\pi}{N}\,uk\right) \quad 0 \leqslant u \leqslant N-1, \tag{2.3}$$

а для двухмерного

$$F(u, v) = \sum_{k=0}^{N-1} \sum_{l=0}^{N-1} f(k, l) \exp\left(i\,\frac{2\pi}{N}\,(ku+lv)\right) \quad 0 \leqslant u, v \leqslant N-1. \tag{2.4}$$

пользуется термин *спектр.* Указанное преобразование можно также определить для функций, аргументом которых является целое число, принимающее значения от 0 до $N-1$. Подобные функции можно рассматривать в качестве выборки N значений некоторого непрерывного сигнала, что, однако, не обязательно должно иметь место в действительности. Сугубо формально для одномерного случая получаем:

Уравнения (2.3) и (2.4) обычно называют *дискретным преобразованием Фурье* (ДПФ). Пределы суммирования у каждой из двух сумм могут быть различными, что позволяет определить ДПФ на прямоугольной области. Нельзя, однако, определить двухмерное преобразование Фурье на области произвольной формы. Если ограничить пределы интегрирования в уравнении (2.2) или пределы суммирования в (2.4), то результат преобразования будет зависеть не только от значений функции f, но и от формы области суммирования/интегрирования[1]. *Дискретное обратное преобразование Фурье (ДОПФ)* определяется следующим образом:

$$f(k) = \frac{1}{N} \sum_{u=0}^{N-1} F(u) \exp\left(\mathrm{i}\,\frac{2\pi}{N}\,uk\right) \qquad 0 \leqslant k \leqslant N-1, \tag{2.5}$$

$$f(k, l) = \frac{1}{N^2} \sum_{u=0}^{N-1} \sum_{v=0}^{N-1} F(u, v) \exp\left[\mathrm{i}\,\frac{2\pi}{N}\,(uk + vl)\right]$$

$$0 \leqslant k, l \leqslant N-1. \tag{2.6}$$

Уравнения, определяющие преобразование, можно записать многими способами, некоторые из которых помогают вводить преобразования иных видов, а также определять эффективные в вычислительном смысле способы их реализации. Обозначим показательную функцию $\exp\left(-\mathrm{i}\frac{2\pi}{N}\right)$ через z и определим матрицу Z:

$$Z = \begin{bmatrix} 1 & 1 & 1 & \cdot & 1 \\ 1 & z & z^2 & \cdot & z^{N-1} \\ 1 & z^2 & z^4 & \cdot & z^{2N-1} \\ \cdot & \cdot & \cdot & \cdot & \cdot \\ 1 & z^{N-1} & z^{2N-1} & \cdot & z^{(N-1)(N-1)} \end{bmatrix} \tag{2.7}$$

Другими словами, $Z_{uk} = z^{uh}$. В большинстве прикладных задач N не только является четным числом, но равно также степени числа 2. Матрицу Z можно упростить, если учесть, что $2^N = 1$, $z^{N/2} = -1$, $z^{N/4} = -\mathrm{i}$ и $z^{3N/4} = \mathrm{i}$. Отметим также, что данная матрица — симметрическая, а скалярное произведение любого из ее столбцов (или строк) и некоторого другого столбца (или строки) равно нулю. Скалярное произведение любого из ее столбцов (или строк) с самим собой равно N. Таким образом, матрица Z обладает тем важным свойством, что ее обратная матрица равна ее сопряженной транспонированной матрице, умноженной на $1/N$. Матрица вида $1/\mathrm{sqrt}(N)\ Z$, у которой обратная матрица равна ее сопряженной транспонированной матрице, называется *унитарной*. Обозначим через f вектор-столбец, компонентами которого служат значения $f(k)$, а через F — вектор-столбец, компонентами которого служат

[1] Можно показать, что соответствующий интеграл равен комплексной свертке фурье-образа функции f с характеристической функцией области интегрирования.

значения $F(u)$. Теперь уравнение (2.3) можно записать в матричном виде:

$$\mathbf{F} = \mathbf{Zf}. \tag{2.8}$$

Обращаясь к двухмерному случаю, отмечаем, что преобразование Фурье эквивалентно умножению всех столбцов функции $f(k, l)$ на матрицу Z, за которым следует умножение справа всех строк функции на Z' (результат транспозиции матрицы Z). В результате уравнение (2.4) принимает вид

$$\mathbf{F} = \mathbf{ZfZ'}. \tag{2.9}$$

Выражения для f и F представляют матрицу изображения и матрицу, получаемую в результате преобразования соответственно. Аналогичные выражения можно получить и для обратного преобразования, если заменить матрицу Z ее комплексно-сопряженной транспонированной матрицей Z*, взятой $1./N$ раз.

Преобразования других видов можно определить, воспользовавшись уравнениями (2.8) и (2.9) и изменив задание матрицы Z таким образом, чтобы 1/sqrt(V) Z оставалась унитарной матрицей. Таким образом можно, в частности, определить *преобразование Адамара*. Матрицы для этого преобразования можно рекурсивно задать следующим образом:

$$Z_2 = \begin{bmatrix} 1 & 1 \\ 1 & -1 \end{bmatrix}, \tag{2.10а}$$

$$Z_{2V} = \begin{bmatrix} Z_N & Z_N \\ Z_N & -Z_N \end{bmatrix}. \tag{2.10б}$$

Таким образом, преобразование Адамара соответствует разложению по периодическим функциям прямоугольной формы в отличие от преобразования Фурье, предусматривающего разложение по синусоидальным и косинусоидальным компонентам. Более того, матрицу Z можно задавать в виде, аналогичном выражению (2.7), положив Z = - 1 и взяв другое правило изменения показателей степени. Главным преимуществом преобразования Адамара является замена умножений комплексных величин изменениями знаков — существенное обстоятельство для первого периода развития вычислительной техники, когда арифметические устройства с плавающей запятой были малодоступны. Во всех остальных отношениях это преобразование позволяет получать точно такие же результаты, как и преобразование Фурье.

Эти преобразования находят применение, главным образом, при конструировании фильтров, предназначенных для улучшения качества изображений, а также при реализации некоторых методов сжатия данных. Они составляют также основу восстановления изображений по проекциям (см. гл. 5). Использование преобразований для сжатия данных оправданно в случаях, когда большая часть компонентов $F(u)$ равна нулю. При выполнении этого условия можно воспользоваться уравнением (2.5) или (2 6)

для восстановления значений $f(k)$ по меньшему, чем N, числу значений фурье-образа. Однако пользователь должен отдавать себе отчет в том, что для описания каждого элемента фурье-образа может потребоваться больше битов, чем необходимо для представления значений исходной функции. В результате коэффициент сжатия может оказаться не так велик, как ожидалось.

Использование определяющих уравнений для реализации преобразования, заданного уравнением (2.8), требует выполнения N^2 вычислительных операций в одномерном случае и N^4 — в двухмерном. Более эффективный алгоритм — *быстрое преобразование Фурье (БПФ)* — можно получить, перегруппировав члены суммы в уравнении (2.3) и использовав то обстоятельство, что произведение ux принимает одни и те же значения при различных значениях u и k. Это преобразование описано в приложении 2.А, причем показано, что в одномерном случае для его реализации требуется всего лишь $N*log_2N$ операций. Алгоритм БПФ можно использовать для получения двухмерного фурье-образа: сначала определяется образ каждой строки изображения, а затем — для всех столбцов. Этот метод иллюстрируется алгоритмом 2.1, в котором процедура БПФ *(N, x)* обеспечивает замену N-элементной матрицы x ее дискретным фурье-образом.

Алгоритм 2.1. Двухмерное БПФ

1. For $l = 0$ до N—1 do: Begin.
2. Запись $f(*, l)$ в матрицу x.
3. Вызов БПФ (N, x).
4. Замена $f(*, I)$ на x.
   End.
5. For k=0 до N - 1 do:
   Begin.
6. Запись $f(k, *)$ в матрицу x.
7. Вызов БПФ (N, x).
8. Замена $f(k, *)$ на x.
   End. 9 Конец

алгоритма.

Каждое обращение к процедуре БПФ *(N, x)* требует затраты вычислительных усилий, пропорциональных $N \log_2 N$. Поскольку общее число таких вызовов равно $2N$, вычислительная сложность в целом имеет порядок $N^2 log_2 (N^2)$.

2.3. ВЫБОРКА

Чтобы обрабатывать на ЭВМ изображение или любой иной сигнал, мы должны преобразовать его в конечный набор чисел. Выборка представляет собой выделение множества дискретных точек во временном или пространственном континууме. В процессе дальнейшей обработки будут использоваться только значения сигнала, зафиксированные в выделенных точках.

Основополагающим математическим результатом для одномерного случая служит теорема Шеннона об отсчетах[1], сформулированная в терминах спектра сигнала. Пусть $d(t)$—дискретное представление $f(t)$, образованное неквантованными импульсами, раз-

$$d(kT) = f(kT),\ k = 0, 1, \ldots, \infty; \tag{2.11а}$$

$$d(t) = 0, \text{ если } t \neq kT. \tag{2.11б}$$

В таком случае можно показать, что $D(w)$ — фурье-образ $d(t)$ связан с $F(w)$ следующим соотношением:

$$D(w) = \sum_{n=-\infty}^{\infty} F\left(w - \frac{2\pi n}{T}\right). \tag{2.12}$$

деленными интервалами длиной T единиц, т. е.

Другими словами, фурье-образ дискретного представления $d(t)$ получается в результате суммирования фурье-образов $f(t)$, взятых со сдвигом. Отметим, что соотношение (2.12) не имеет отношения к дискретному фурье-образу $d(t)$; оно характеризует непрерывное преобразование некоторой функции, принимающей ненулевые значения лишь в определенные дискретные моменты времени. Различие результатов преобразования функций d и f можно уяснить, обратившись к примерам, приведенным на рис. 2.1. На верхнем левом графике $F_1(w)=0$ при $w>Pi/T$, в то время как на верхнем правом графике это не так. Графики $D(w)$ для этих двух случаев приведены в средней и нижней частях рис. 2.1. В первом случае, если $D(w)$, задан, то $F(w)$ можно определить точно и, следова-

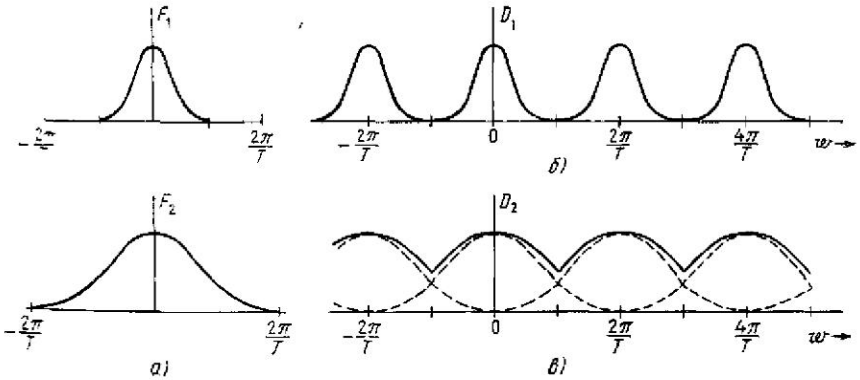


Рис 2 1. Фурье-преобразование дискретизованных сигналов.
alfa-спектры $F_1(n)$ и $F_2(w)$ недисиретизованных сигналов $f_1(t)$ и $f_2(t)$ соответственно; *б* — спектр $D_1(w)$ дискретизованного сигнала $d_1(t)$; *в* — спектр $D_2(u)$ - - - очертания фурье-образа непрерывного сигнала, повторяемого с частотой $2\pi T$

[1]Эта теорема известна в отечественной литературе как теорема Котельникова *Прим. перев.*

тельно, восстановить сигнал $f(t)$ по его дискретному представлению $d(t)$. Это невозможно во втором случае из-за перекрытия членов уравнения (2.12). На основании сказанного шеннонова теорема отсчетов формулируется следующим образом.

Теорема 2.1. Пусть w_{max} — максимальное значение w, при котором фурье-образ $F(w)$ не равен нулю. Тогда сигнал $f(t)$ поддается точному восстановлению по выборочным отчетам, если время между соседними отсчетами меньше, чем π/w_{max} т. е. частота выборки равна, по меньшей мере, удвоенному значению w_{max}

Отметим, что приведенная теорема не предлагает какого-либо способа восстановления непрерывного сигнала по его дискретным отсчетам. Она лишь указывает на возможность такого восстановления. На самом деле, при использовании минимальной частоты выборки для восстановления сигнала приходится обращаться к достаточно сложным методам. Если предпочтительным является определенный тип восстановления, то частота выборки вполне может оказаться выше минимального значения, определяемого теоремой отсчетов.

Пример 2.1. Допустим, что восстановленный сигнал состоит из постоянных, равных выборочным значениям (как показано на рис. 22) Пусть $f(t) = Esin(wt)$ при некотором значении w. Если мы хотим, чтобы максимальное

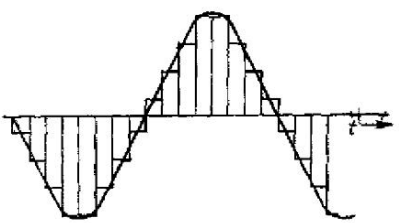


Рис. 2.2. Восстановление сигнала с помощью кусочно-постоянной аппроксимации

отклонение восстановленного сигнала от исходного не превышало заданной величины e, то, как следует из рис. 2.2, должно выполняться следующее соотношение:

$$|f(kT \pm T/2) - f(kT)| < e, \qquad (2.13)$$

где k — момент к T — интервал выборки. Далее с помощью теоремы о среднем можно показать, что это неравенство выполняется, если максимум абсолютной величины $f(t)$—произвольной $f'(t)$, меньше, чем $2e/T$. Для конкретного сигнала $f(t)$ можно также с помощью тригонометрической формулы вычислить максимальное значение разности в неравенстве (2.13) и показать, что оно эквивалентно неравенству

$$E\,|\sin(\omega T/2)| < e.$$

Если значение синуса малого угла аппроксимировать значением собственно угла, то последнее неравенство принимает вид

$$T < 2e/(E\omega),$$

и граница, определяемая этим неравенством, является существенно меньше шенноновской. К такому результату приводит и использование теоремы о среднем.

2.3.2. ВЫБОРКА В ДВУХМЕРНОМ СЛУЧАЕ

В обработке изображений выбор типа восстановления обычно ограничен и это следует всегда иметь в виду при использовании обобщения теоремы отсчетов на двухмерный случай. Несмотря на то что теорема Шеннона допускает тривиальное обобщение на двухмерный случай, последнее не представляет особого интереса вследствие ограниченных возможностей известных алгоритмов восстановления. Таким образом, при работе с изобразительными данными частота выборки обычно должна быть существенно выше той, которая определяется результатами спектрального анализа. В частности, можно проводить выборку с частотой, соответствующей верхней границе, которая определяется соотношением вида (2.14). Полученный таким способом интервал выборки будет равен значению, определяемому теоремой отсчетов и умноженному на коэффициент *2e/(E*Pi)*. Если отношение *e/E* составляет 1%, то выборку следует осуществлять с частотой в 157 раз превышающей предельное значение частоты выборки, указываемое теоремой Шеннона!



Рис. 2.3. Дискретизованное изображение, содержащее 256х х256 выборочных элементов и 256 уровней серого тона

Эту проблему иллюстрируют рис. 2.3—2.5. На первом из них представлено изображение, обладающее высокой четкостью (256X256 элементов изображения) и воспринимаемое большинством смотрящих на него как аналоговое. Два других изображения получены из этого изображения при помощи выборки с пропусками: изоб-



Рис. 2.4. Дискретизованное изображение, содержащее 64х64 выборочных элементов и воспроизведенное на большом экране

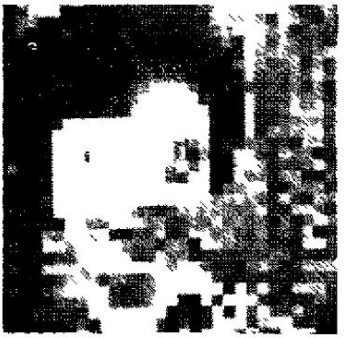


Рис. 2.5. Дискретизованное изображение, содержащее 32х32 выборочных элементов и воспроизведенное на большом экране

ражение на рис. 2.4 содержит 64x64 выборочных элементов, а изображение на рис. 2.5 — 32x32 выборочных элементов. Эти изображения воспроизведены на большем экране, что стало возможным благодаря многократному воспроизведению значения каждого отсчета— 16 и 64 раз соответственно. Совершенно очевидно, что качество обоих изображений низкое, однако причиной этого служит не только неполнота выборки. Вы можете попробовать взглянуть на эти изображения с некоторого расстояния или прищурившись. Качество изображений повысится, поскольку в них содержится наибольшая часть информации. Явные искажения вносит именно кусочно-постоянное восстановление. Можно получить алгоритмы, осуществляющие линейную интерполяцию между значениями выборочных элементов (см. задачу 2.3), и в результате при воспроизведении изображения с низкой четкостью на экране с высокой разрешающей способностью дополнительным пикселам присваиваются промежуточные относительно элементов выборки значения. Однако процесс интерполяции может протекать крайне медленно и потому его использование при работе со стандартным дисплеем непрактично. Лишь после разработки графических дисплеев, которые будут осуществлять такую интерполяцию локально (с помощью специализированного устройства), станет возможной дискретизация изображений с частотой, более близкой к значению, определяемому теоремой Шеннона. (В терминах цифровой фильтрации эти устройства эквивалентны элементам *синхронизации высшего порядка.)*

Необходимость прибегать к избыточной выборке часто в определенной мере препятствует созданию чисто цифровой системы обработки изобразительной информации. Один из возможных компромиссов заключается в смешении аналогового и дискретного выходных сигналов с помощью телевизионной аппаратуры. Сигналы поступают в видеоконтрольное устройство от двух источников: контроллера растрового графического устройства и телевизионной передающей камеры или ленты для аналоговой видеозаписи. Эти сигналы можно смешать, организовав наложение, скажем, статистической информации (полученной дискретными средствами) на (аналоговую) карту города [1].

Назначение выборочных точек не вызывает затруднений в случае одномерных сигналов, но в случае двухмерных возникают серьезные проблемы. Теперь дадим формальное определение нескольким понятиям, которыми будем часто пользоваться.

Определение 2.1. Пусть P — плоскость, содержащая аналоговое изображение. *Элементом дискретизации* называется компактное выпуклое подмножество P, по которому вычисляется значение выборочного элемента, входящего в состав дискретизированного изображения. Объединение таких элементов называется *сеткой (выборки).* Отдельный отсчет — выборочный элемент — называется

[1] Об этом приеме мне стало известно от доктора Жана-Поля Джейкоба из фирмы IBM Research Laboratories.

элементом изображения, или *пикселом.* Пусть D — плоскость визуального отображения; пикселы в этом случае отображаются на элементы воспроизведения изображения и в результате восстанавливается аналоговое изображение.

Следует подчеркнуть, что мы не предполагаем отсутствие пересечений элементов дискретизации. Действительно, в большинстве устройств дискретизации они частично перекрываются, хотя элементы воспроизведения изображения обычно не пересекаются. На рис. 2.6 представлены типичная сетка выборки и соответствующее

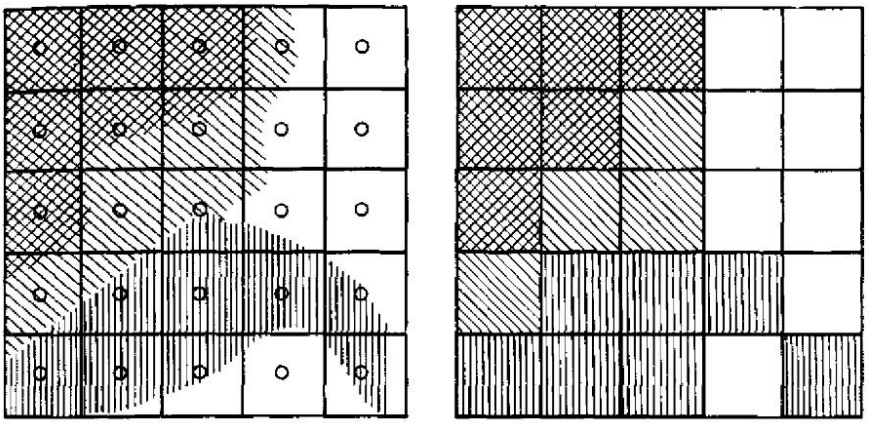


Рис. 2 6. Выборка в двухмерном случае и соответствующее отображение.

Обратите внимание на появление на отображении резких границ, они соответствуют высоким частотам, внесенным, как это видно на рис 2 1 выборкой Эта иллюстрация также хорошо объясняет причины низкого качества изображений, приведенных на рис 24 к 25

отображение. Хотя использование разных сеток для выборки и отображения теоретически возможно и иногда практически неизбежно, делать этого не рекомендуется из-за возникающих при этом искажений изображения. Для характеристики большинства процедур дискретизации используется *передаточная функция точки,* g*(r),* отражающая вклад различных точек в значение выборочной точки в зависимости от их расстояния r от нее. Функция $g(r)$ является убывающей функцией расстояния r, равной нулю за пределами элемента дискретизации. Если функция $g(r)$ убывает достаточно быстро, то можно допускать отсутствие перекрытия элементов Дискретизации и в качестве первого приближения обозначать термином «элемент» как элементы дискретизации, так и элементы воспроизведения изображения.

В обработке изображений чаще всего используется *квадратная сетка*, состоящая из элементов-квадратов, упорядоченных в виде шахматной доски. *Шестиугольная сетка* (рис. 2.7) часто обсуждается в литературе, но редко используется. В то время как в теореме отсчетов для одномерного случая речь идет лишь о размерах

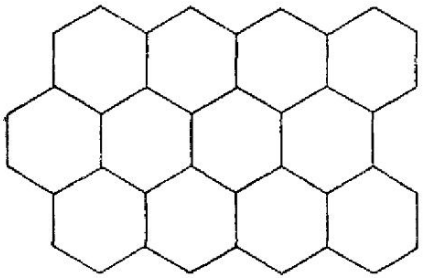


Рис. 2 7. Шестиугольная сетка.

Ее преимущество закчючается в том что все элементы, соседние с элементом *С*, примыкают к нему одним и тем же способом а в случае квадратной сетки следует различать соседние элементы, примыкающие к сторонам или углам, недостатком шестиугольной сетки является различие расстояний между центрами элементов по вертикали и горизонтали

интервала выборки, при определении размеров элементов сетки всегда необходимо учитывать их форму и вид передаточной функции точки Ниже мы будем иметь дело, главным образом, с квадратными сетками, а к сеткам иной формы будем обращаться лишь эпизодически

2.4. ИСКАЖЕНИЯ ИЗОБРАЖЕНИЙ ВСЛЕДСТВИЕ НЕДОСТАТОЧНО ВЫСОКОЙ ЧАСТОТЫ ВЫБОРКИ

Если интервал выборки не удовлетворяет условиям теоремы 2 1, то, как это было показано на нижнем графике рис. 2 1, возникают спектральные искажения. Высокие частоты свертываются с низкими, порождая явление, называемое *искажением изображения вследствие недостаточно высокой частоты выборки.* Возникновение такого рода искажении в одномерном случае иллюстрирует рис 2 8 высокочастотный сигнал после дискретизации со слишком

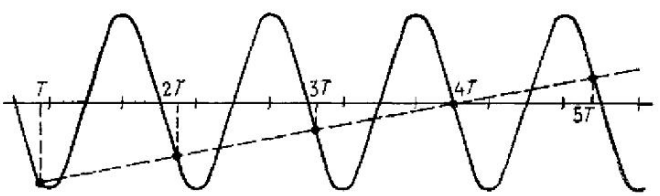


Рис. 2.8 Пример возникновения искажений вследствие недостаточно высокой частоты выборки.

малой частотой приобретает вид низкочастотного Типичным примером такой ситуации, возникающей при обработке изобразительной информации, является дискретизация изображения, содержащего текст Если отсчеты делаются слишком редко, то результат имеет вид хаотической комбинации темных и светлых участков, а не очертаний букв. Другим примером служит дискретизация изображения, на котором воспроизведение уровня серого тона достигается изменением плотности черных и белых точек. Если частота

выборки соизмерима с размером точек, то выборочные значения будут соответствовать белому или черному, однако их распределение будет отличаться от распределения белых и черных точек на исходном изображении. В результате, качество дискретизированного изображения может оказаться очень низким. Поэтому при обработке текста требуется увеличение частоты выборки. Во втором случае следует обратиться к иным средствам: усреднять значения точек обрабатываемого изображения. Другими словами, необходимо пропустить сигналы перед началом дискретизации через низкочастотный фильтр.

Пример 2.2. Допустим, нас интересуют лишь частоты ниже 1000 Гц, а сигнал содержит «неинтересную» составляющую с частотой 1800 Гц При выборке с частотой 2000 Гц появляется ложная низкочастотная составляющая (200 Гц) в спектре дискретизованного сигнала, которая порождена составляющей с частотой 1800 Гц. Чтобы избавиться от этой ложной составляющей, прежде чем приступать к дискретизации исходного сигнала, ее следует пропустить через *аналоговый низкочастотный фильтр,* обеспечивающий подавление всех частот, превышающих 1000 Гц.

Искажения, возникающие на изображениях вследствие недостаточно высокой частоты выборки, порождают при их обработке серьезные проблемы, связанные с построением аналоговых фильтров для предварительной обработки сигналов. Расфокусировка оптической системы устройства дискретизации изображения эквивалентна пропуску изображения через простой низкочастотный фильтр — в некоторых случаях достаточно применения этого приема

В литературе, посвященной машинной графике, искажениями, появляющимися вследствие недостаточно высокой частоты выборки, часто называют искажения, возникающие в результате кусочно-постоянного восстановления (подобные искажения представлены на рис. 24 и 25). Строго говоря, проблема здесь заключается не в отображении некоторой высокочастотной составляющей на некоторую низкочастотную составляющую, а в появлении дополнительных высокочастотных составляющих, порождаемых именно используемым методом восстановления. Путаница возникает вследствие того, что одним из методов разрешения этой проблемы является увеличение частоты выборки (см. пример 2 1) и этот же метод используется для устранения искажений изображения, возникающих вследствие недостаточно высокой частоты выборки. Разница между этими двумя явлениями становится очевидной если учесть, что вид изображений, приведенных на рис. 24 и 2.5, улучшен при помощи расфокусировки, т. е. низкочастотной фильтрации *цифрового изображения. С* помощью подобной фильтрации можно устранить искажения, вызванные недостаточно высокой частотой выборки, только в случае, если она применяется к *аналоговому изображению.*

2.5. КВАНТОВАНИЕ

Значения, полученные в процессе дискретизации аналогового сигнала, должны быть представлены с помощью конечного числа 5итов, определяемого возможностями используемой ЭВМ. Этот процесс, называемый *квантованием,* можно рассматривать как отображение действительных чисел в область целых чисел. Выбор числа уровней серого тона для представления черно-белых изображений должен проводиться с учетом особенностей зрения человека, однако обсуждение этой темы выходит за пределы задач нашей книги. Очевидно вполне правильно считать, что для представления большинства изображений достаточно 8 бит (256 уровней), хотя часто и 6 бит позволяют обеспечить хорошее воспроизведение. Для обеспечения хорошего качества воспроизведения цветных изображений могут потребоваться значительные затраты: по 6 бит (64 уровня) на каждый из трех основных цветов (всего 18 бит) или даже больше. Во многих прикладных задачах могут быть достаточными затраты по 4 бита (16 уровней) на один цвет. Минимально допустимому качеству соответствуют затраты одного байта на представление одного пиксела, причем для передачи двух цветов используется по 3, а для передачи третьего — 2 бита. На рис. 2.9 и 2 10 приведены примеры черно-белых изображений, подвергнутых квантованию с различным числом уровней. Сопоставление обоих изображений с оригиналом, при представлении которого затрачивалось по 8 бит на пиксел (см. рис. 2.3), показывает, что они выглядят хуже оригинала, главным образом из-за видимых контуров, разделяющих уровни серого тона. Существенная тоновая информация сохраняется даже на тех изображениях, где для передачи уровней серого тона отводится 2 бита. Качество изобра



Рис 29 Вариант изображения, приведенного на рис 2 3, полученный в результате квантования с использованием восьми уровней серого тона (трехбитовое изображение)



Рис 2 10 Вариант изображения, приведенного на рис. 2 3, полученный в результате квантования с использованием четырех уровней серого тона (двухбитовое изображение)

жения, характеризуемого шестнадцатью уровнями серого тона (4-битовое изображение), столь высоко, что при воспроизведении (печати) обоих изображений на бумаге оно неотличимо от оригинала. (Тем не менее, о наличии различий можно говорить при одновременном воспроизведении обоих изображений рядом на экране видеоконтрольного устройства.)

Улучшить вид изображений, подвергнутых грубому квантованию, можно добавлением *низкочастотного псевдослучайного шума:* перед квантованием к функции *f(x, y),* представляющей изображение, в точке *(x, y)* прибавляется низкочастотный шум *d(x, y).* Для получения этого шума можно использовать случайный процесс или определить его соответствующее значение, исходя из положения точки *(x, y).* В любом случае это значение никак регулярным образом не связано с *f(x, y).* Этот процесс обычно приводит к исчезновению контуров и в результате, несмотря на наложение на изображение шума, его общий вид улучшается. На рис. 2.11 и 2.12 приведены размытые вследствие добавления низкочастотного шума варианты изображений рис. 2.9 и 2.10 соответственно. В качестве значения шумовой составляющей *d(x, y) с* равной вероятностью берется одно из пяти следующих чисел:

$$-2^{(6-b)},\ -2^{(5-b)},\ 0,\ 2^{(5-b)} \text{ и } 2^{(6-b)},$$

где *b* - число битов, подлежащих отображению. В качестве значения пиксела берутся *b* старших битов соответствующей суммы. (Если сумма оказывается отрицательной, то перед наложением разрядной маски она приравнивается нулю.)

Выбор соотношения между частотой выборки и числом уровней квантования также представляет определенную проблему, однако до сих пор ей уделялось мало внимания. Если для хранения заданного изображения можно использовать лишь фиксированное



Рис. 211. Вариант изображения, приведенного на рис. 2 9, полученный в результате наложения на последнее низкочастотного псевдослучайного шума

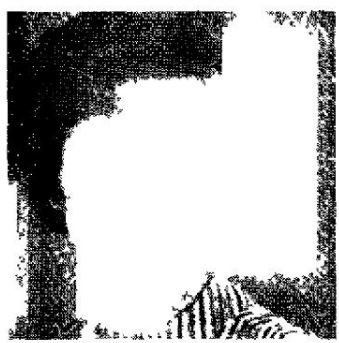


Рис 2 12. Вариант изображения, приведенного на рис. 2.10, полученный в результате наложения на последнее низкочастотного псевдослучайного шума

число битов, то как их лучше распределить? Так, 8192 бит можно использовать для хранения сетки 64x64 (4096 пикселов), предусмотрев 4 уровня серого тона (2 бита), или для хранения сетки .32x32 (1024 пиксела), предусмотрев 256 уровней серого тона (8 бит). Очевидно, выбор определяется типом изображения, но, кроме того, при работе с конкретным изображением можно для отдельных его частей использовать различные частоты выборки и число уровней квантования. Несколько задач такого рода рассмотрено в разд. 3.2. Другим ограничением является объем необходимых вычислений. При решении большинства задач вычислительные затраты в определенной степени зависят от числа выборочных элементов (пикселов), но на них не отражается число битов, затрачиваемых на представление одного пиксела, если это число не превышает длину слова используемой ЭВМ. Таким образом, время обработки изображения, состоящего из 4096 пикселов, на каждый из которых затрачивается по одному биту, может оказаться в 4 раза больше времени обработки изображения, состоящего из 1024 пикселов, на каждый из которых затрачивается по 4 бита.

Большинство работ, отраженных в литературе, носит экспериментальный характер; в библиографических замечаниях приведено несколько соответствующих ссылок. Следующий пример иллюстрирует несколько специфических проблем, возникающих в связи с только что изложенным.

Пример 2.3. Задача заключается в дискретизации двухуровневых изображений типа страниц книжного текста. Характер функции рассеяния устройства дискретизации приводит, однако, к получению выборочных элементов, принимающих значения в обширном диапазоне. (Можно допустить, что значение выборочного элемента равно среднему значению входного сигнала, вычисленному по элементу дискретизации.) Это означает, что нам приходится работать с изображением класса 1, хотя на вход подается изображение класса 2. Мож-,но попытаться возвратиться к изображению класса 2, затрачивая на каждый

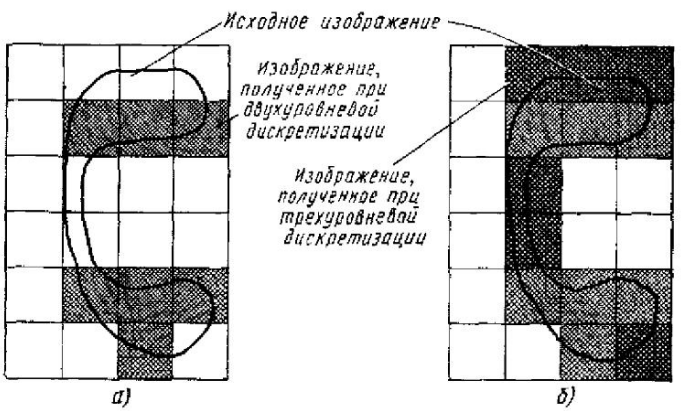


Рис. 2.13. Иллюстрация к примеру 2.3:

а — появление на изображении разрыва вследствие недостаточного разрешения; *б* — обеспечение правильной передачи топологии символа благодаря увеличению числа уровней серого тона

пиксел только по одному биту и приравнивая его нулю, если входной сигнал оказывается меньше некоторого порогового, и единице в противном случае. Очевидно, пикселы, соответствующие тем элементам дискретизации, которые расположены вблизи границ областей исходного изображения, получают значения 0 и 1 более или менее произвольным образом; это может привести к появлению или исчезновению промежутков, соответственно отсутствовавших или имевшихся на исходном изображении, как показано на рис. 2.13,а. Поэтому элементы дискретизации можно выбирать достаточно малыми, чтобы даже в наиболее узких частях области элемент дискретизации целиком попадал в участок одного цвета. (Дополнительные сведения можно найти в гл. 7.) С другой стороны, можно ввести 4 уровня квантования (по 2 бита на каждый пиксел) при более низком разрешении, как показано на рис. 2.13,б).

2.6. БИБЛИОГРАФИЧЕСКИЕ ЗАМЕЧАНИЯ

Использование преобразований и процедур линейной фильтрации подробно рассматривается в большинстве книг, посвященных обработке изображений. Книга Прэтта [2.9] является хорошим источником информации по проблемам, которым посвящена эта и следующая главы; она рассчитана на более подготовленного читателя. В ней много внимания уделено обработке сигналов и хорошо представлены оптические аспекты обработки изображений. Книга Каслмана [2.1] лаконичнее, но она охватывает более узкий круг вопросов. В работах [2.3, 2.11] рассматриваются многие из этих проблем, а также сегментация и некоторые аспекты распознавания образов[1]. Значительное число работ посвящено практическим аспектам квантования и дискретизации при использовании дискретных изображений. Ноултон и Хармон подготовили интересный обзор ранних работ в этой области [2.6]. Восприятие изображений, подвергнутых «грубой» дискретизации (подобно изображению, приведенному на рис. 2.5), обсуждается в работе [2.4] [2]. Некоторые аспекты выравнивания изображений при помощи наложения низкочастотного псевдослучайного шума рассматриваются в работе [2.5]. Свежий обзор методов кодирования изображений, посвященный дискретизации и квантованию, содержится в работе [2.8]. Наиболее глубокое теоретическое рассмотрение проблемы кодирования проведено в работах, включенных в сборник [2.10], в частности, интересна статья [2.12], посвященная зрительной системе человека. В литературе по машинной графике много внимания уделяется использованию результатов теории вы-

[1] В 1982 г. было выпущено второе существенно расширенное и перерабо танное издание книги Rosenfeld A. and Kak A. C. Digital Picture Processing, Second Edition, New York: Academic Press, 1982, v 1 — 454 p.; v.2 — 368 p., (Computer Science and Applied Mathematics Series of Monographs and Tex tbooks). — *Прим. перев.*

[2] Далее, опираясь на эти концепции и. возможно, под влиянием работы Хармона и Джулеса, написал картину, состоящую из прямоугольников, в пре делах которых цвет приблизительно, но не точно, однороден. Человек, рассмат ривающий картину с некоторого расстояния, видит лишь прямоугольники и воспринимает ее как «грубо» дискретизированный портрет Линкольна. При приближении к картине становятся различимыми детали прямоугольников и в окне появляется изображение женщины.

борочного метода для надлежащего воспроизведения изображений. Сравнительно недавно появились работы, содержащие интересные теоретические результаты [2.2, 2.7].

2.7. ЗАДАЧИ

2.1. Реализуйте алгоритм 2.1, исходя из того, что размеры быстродейст вующей памяти достаточны для запоминания изображения.
2.2. Решите задачу 2.1 при условии, что изображение не умещается в быст родействующей памяти. В этом случае требуется написать процедуру нахож дения транспонированной матрицы, не требующую запоминания исходной мат рицы в целом как некоторого массива.
2.3. Напишите программу линейного кусочно-постоянного восстановления *(синхронизация первого порядка)* дискретизованного изображения. При этом можно воспользоваться следующей формальной моделью. Если a, b, c и a — значения четырех пикселов, образующих квадрат

$a\ b$
$c\ d$

x и y — координаты относительно левого верхнего угла, h—длина стороны квадрата, то значения z промежуточных точек можно определять с помощью следующих выражений:

$$z = \frac{y(c-a) + x(b-a)}{h} + a, \quad x < h - y;$$

$$z = \frac{y(d-b) + x(d-c)}{h} + c + b - d, \quad x \geqslant h - y.$$

[Эти выражения известны для элементарной геометрии; кроме того, их можно получить из соотношения (162).]
2.4. Воспользуйтесь программой из задачи 2.3 для изучения возможностей выбора соотношения частоты выборки и числа уровней квантования. (Писать эти программы можно, также исходя из того, что они должны служить со ставными частями программы-редактора изображений; см. следующую задачу.)
2.5. Усовершенствуйте программу-редактор изображений, написанную в со ответствии с заданием задачи 1.5, оснастив ее средствами, обеспечивающими возможность исследовать влияние изменений частоты выборки, числа уровней квантования и их соотношения.
Рекомендации. Можете предусмотреть в редакторе команду РАСТЯЖЕНИЕ и для изучения влияния изменений частоты выборки использовать подмножество пикселов исходного изображения, рассматривая их как новое изображение, которое затем воспроизводится с помощью процедуры растяжения, с теми же размерами, что и исходное изображение. С другой стороны, можете в меню световых кнопок добавить новые команды: ВЫБОРКА и УРОВЕНЬ. Первая команда обеспечивает выборку каждого второго пиксела по вертикали и горизонтали (одного из четырех) и воспроизведение результата при помощи операции растяжения. Команда УРОВЕНЬ обеспечивает пропуск («зачернение») битов, начиная с младшего. Кроме того, можете предусмотреть команды, допускающие числовые аргументы, которые точно определяют частоту выборки и число уровней квантования.
2.6. Проанализируйте схему наложения на изображение низкочастотного псевдослучайного шума, рассмотренную в разд. 2.5.

Приложение 2.А. Быстрое преобразование Фурье

Кратко рассмотрим БПФ. Более подробные сведения читатель может найти в литературе по обработке сигналов. В качестве отправного пункта восполь-

зуемся уравнением (2.3), заменив в нем показательную функцию на z, как это сделано в разд. 2.2:

$$F(u) = \sum_{k=0}^{N-1} f(k)\, z^{uk}, \quad 0 \leqslant u \leqslant N \quad 1. \tag{2 A.1}$$

Введем новый индекс суммирования m, который пробегает значения от О до $N/2$—1 и связан с прежним индексом суммирования k следующими соотношениями:

$k = 2m$, если k — четное; (2.A.2a)

$k = 2m + 1$, если k — нечетное. (2.A.2б)

Тогда уравнение (2.A.1) можно переписать следующим образом:

$$F(u) = \sum_{m=0}^{(N/2)-1} [f(2m)\, z^{2mu} + f(2m+1)\, z^{(2m+1)u}]. \tag{2.A.3}$$

Пусть далее $M = N/2$ и $g = \exp(-i\frac{1\pi}{M})$, отмегим, что $z^2 = g$. В таком случае уравнение (2 A.3) можно переписать

$$F(u) = \sum_{m=0}^{M-1} f_e(m)\, g^{um} + z^u \sum_{m=0}^{M-1} f_0(m)\, g^{um}, \tag{2.A.4}$$

где f_e и f_0 — новые функции, полученные в результате разделения точек, соответствующих четным и нечетным значениям аргумента. Две суммы, входящие в уравнение (2 A.4), очень сильно напоминают фурье-преобразования функций, заданных M отсчетами, с тем отличием, что в формулах, определяющих эти фурье-преобразования, индекс суммирования и пробегает значения от 0 до N—1 Обратим внимание, что

$$F(u + N) = F(u), \tag{2.A 5}$$

поскольку $\exp(-i\, N\frac{2\pi}{N}) = 1$ Таким образом, по M значениям фурье-образа функции f_e (или f_0) можно тривиально найти N значений. Следовательно, для вычисления фурье-преобразования справедливо следующее *рекуррентное* соотношение:

$$F(u) = F_e(u) + \exp\left(-i\frac{2\pi u}{N}\right) F_0(u). \tag{2.A.6}$$

Соотношение (2.A 6) является основой БПФ. Во-первых, соответствующая вычислительная процедура легко программируется. Запишем ее в виде алгоритма 2.A.1.

Алгоритм 2.A. 1. Быстрое преобразование Фурье

Процедура БПФ (N, f)

1. If N равно 2, then do:
   Begin.
2. Замена f(0) на f(0)+f(1) и f(1) на f(0)—f(1).
3. Return.
   End
4. Else do:
   Begin.
5. Определите функцию g как состоящую из всех отсчетов функции f с четным индексом и функцию h как состоящую из остальных отсчетов функции f.
6. Call procedure *БПФ (N/2, g).*
7. Call procedure *БПФ (N/2, h).*

8. Замена *f(t)* на *g(t)* +exp(—i 2π / *N)h(t)for t=0* до *N—1.*
End.
9. КОНЕЦ ПРОЦЕДУРЫ

Процедура предусматривает обращение к самой себе при *N*>2, а при *N=2* — использование следующих формул:

$$F(0) = f(0) + f(1); \qquad (2.A.7a)$$

$$F(1) = f(0) - f(1). \qquad (2.A.7б)$$

Совершенно очевидно, что последовательное деление *N* пополам можно реализовать лишь в случае, когда *N* есть некоторая степень числа 2, и только для таких значений существует простой алгоритм БПФ. Кроме того, уравнением (2.А.6) можно воспользоваться для определения трудоемкости вычислительной процедуры. Пусть *C(N)* — трудоемкость для случая *N* отсчетов. Кроме затрат на вычисление значений двух фурье-образов, входящих в правую часть уравнения (2.А.6), само оно требует затрат, пропорциональных значению *N* и связанных с умножением на показательную функцию и последующим сложением. Если *c* — постоянная, характеризующая трудоемкость этих операций, то C (*N*) определяется следующим уравнением:

$$C(N) = 2C(N/2) + cN. \qquad (2.A.8)$$

Аналогичным образом находим, что

$$C(N/2) = 2C(N/4) + c(N/2) \qquad (2.A.9)$$

$$C(N) = (N/2)c' + cNn, \qquad (2.A.10)$$

где *n* — число таких уравнений; *n,* в частности, равно числу членов ряда 1, 2, 4, ..., *N/4,* которое равно $log_2 N$ минус 1. В таком случае уравнение (2.А.10) принимает вид

$$C(N) = cN \log_2 N + O(N), \qquad (2.A.11)$$

и т. д. Из уравнений (2.А.7) следует, что c(2) эквивалентно трудоемкости двух сложений, которую обозначим через *c'.* Если выписать уравнения, аналогичные (2.А.9), для *C(N /L),* где L=4, ..., *N/4,* умножить каждое из них на *L*, затем сложить с уравнением (2.А.8) и далее с уравнением (2.А.9), умноженным на два, то оказывается, что

второй член в правой части которого представляет величины, линейно пропорциональные *N.*

Глава 3

ОБРАБОТКА ТОНОВЫХ ИЗОБРАЖЕНИЙ

3.1. ВВЕДЕНИЕ

Известны две основные разновидности обработки тоновых изображений: преобразования, не выводящие изображения за пределы соответствующего класса, типа *фильтрации* и *улучшения качества изображения,* и преобразования, переводящие изображения из класса 1 в класс 2, типа *сегментации.* В большинстве методов, обеспечивающих выполнение подобной обработки, прямо или косвенно используются статистические характеристики изображений. Мы рассмотрим две статистические характеристики такого рода: *гистограмму* распределения уровней серого тона (см.разд. 3.2) и *матрицу совместной встречаемости* пар уровней серого тона у пар пикселов (см. разд. 3.3). Применение этих характеристик в процедурах фильтрации обсуждается в разд. 3.4 и 3.5, а в процедурах сегментации — в следующей главе.

Если изображение рассматривается как случайный процесс, то следует определить плотность распределения вероятностей *первого порядка* $p_1(P, Z)$ для события, состоящего в том, что пиксел P (задаваемый его местоположением) характеризуется уровнем яркости (или цветом) Z. При работе с черно-белыми изображениями всегда предполагается в дальнейшем, что яркость Z изменяется от 0 (очень темный пиксел) до некоторого $L>0$ (очень яркий пиксел). Связь вероятности определенного уровня пиксела с его местоположением позволяет использовать эту модель для порождения интересных изображений.

Пример 3.1. Рассмотрим изображение, представляющее собой зачерненный символ на светлом фоне. Пусть Co — множество тех пикселов, которые принадлежат идеальному изображению соответствующего символа и характеризуются яркостью D. Яркость пикселов фона равна L. Вследствие наличия шума при каждом взгляде на символ мы обнаруживаем некоторое новое множество зачерненных пикселов C_i. Пусть C — пересечение всех возможных C_i и B — разность объединения C_i и пересечения C. Можно считать, что при умеренных значениях шума C окажется подмножеством Co, а объединение B к C будет включать Co. Очевидно, что множество B представляет собой пикселы, расположенные вдоль границы Co. Пусть A—часть изображения, не покрытая множествами B и C. (На рис. 3.1,*а* это показано для конкретного символа.) При отсутствии шума C = Co, а B — пустое множество. В таком случае изображение порождается по следующему вероятностному закону.

$p_1(P, Z) = d(Z-D)$, если P принадлежит C,

$p_1(P, Z) = d(Z-L)$, если P принадлежит A,

где $d(x)$ — функция, принимающая значение 1 при нулевом значении аргумента и 0 в противном случае. При наличии шума данное изображение можно воспроизвести с помощью вероятностного закона

$$p_1(P, Z) = f(Z - D), \quad \text{если } P \text{ принадлежит } C,$$

$$p_1(P, Z) = f(Z - L), \quad \text{если } P \text{ принадлежит } A,$$

$$p_1(P, Z) = [f(Z - D) + f(Z - L)]/2, \quad \text{если } P \text{ принадлежит } B,$$

где $f(x)$ —функция, представленная на рис. 3.1,*б*.

Этот пример иллюстрирует также и некоторую ограниченность предложенной модели. Область B будет выглядеть как множество случайных точек, хотя можно полагать, что пикселы, расположенные ближе к C, будут черными, а ближе к A — белыми. Чтобы получить более реалистическую модель, следует использовать также плотность распределения вероятностей *второго порядка* $P2(P, Q, Z, Y)$ для события, состоящего в том, что пиксел P характеризуется яркостью Z и пиксел Q — яркостью Y.

Пример 3.2, Изображение, образованное перемежающимися вертикальными темными и светлыми полосами, можно воспроизвести, задав плотность распределения вероятностей второго порядка $p_2(P, Q, Z, Y)$ следующим образом:

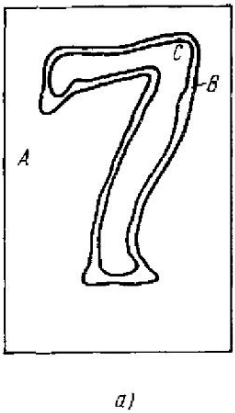
$$p_2(P, Q, Z, Y) = f(Z - Y),$$

если P *и* Q расположены вертикально или горизонтально и расстояние между ними есть четное кратное ширины полосы;

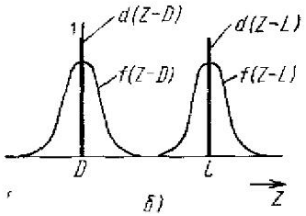
$$p_2(P, Q, Z, Y) = 1 - f(Z - Y),$$

если P и Q расположены горизонтально и расстояние между ними есть нечетное кратное ширины полосы.

Это задание неполно, поскольку оно не определяет вид $p_2(P, Q, Z, У)$ для всех случаев взаимного расположения P и Q, за исключением нахождения их в одной и той же вертикальной или горизонтальной полосе. Полное задание плотности распределения вероятностей второго порядка p_2 потребовало бы значительно больше места



а)



б)

Рис. 3.1. Иллюстрация к примеру 3.1:
а — расположение областей *A, B* и C; *б* — вид используемых функции *f(x)* и *d(x)*, интеграл функций *f(x)* должен быть равен единице при всех значениях *к*

Приведенные примеры подчеркивают одну трудность, возникающую при использовании плотностей вероятностей, особенно второго порядка: если функция задается в табличной форме, то соответствующая таблица весьма велика; если функция задается аналитически, то требуются сложные пояснения. Один из распространенных способов преодоления этого затруднения заключается в пренебрежении зависимостью от P при использовании плотности вероятности первого порядка и учета лишь взаимного расположения P *и* Q при использовании плотности вероятности второго порядка. Более того, для получения оценок этих функций можно ограничиться только одним изображением. Читатель, знакомый с теорией случайных процессов, должен понимать, что подобный подход допустим лишь при предположении об эргодичности процесса, однако это условие не выполняется для большинства нетривиальных процедур порождения изображений. Следовательно, полученные таким образом оценки следует применять осмотрительно.

3.2. ГИСТОГРАММЫ И ИХ ВЫРАВНИВАНИЕ

Алгоритм 3.1 показывает, каким образом вычисляется оценка плотности вероятности первого порядка $p_1(P, Z)$ по одному изображению при условии, что она не зависит от P. Полученный результат называют *гистограммой* изображения $H(Z)$. Если $f(P)$ характеризует уровень яркости некоторого пиксела P, то алгоритм 3.1 можно использовать для оценивания гистограммы $H(Z)$. На рис. 3.2 приведена гистограмма изображения, показанного на рис. 2.3. Обратите внимание на отсутствие некоторых уровней;

они отсутствуют и на изображении, для которого построена гистограмма. Выпадение этих уровней связано с одной технической проблемой, возникшей при дискретизации изображения с помощью АЦП — одному из битов значения не присваивались вообще и в результате некоторые уровни серого тона не были воспроизведены.

Алгоритм 3.1. Оценивание гистограммы

Обозначения: f(P) — значение пиксела *P,* изменяющееся в диапазоне [O, L]. *H* — массив гистограммы.

1. Присвоить элементам массива *H(Z)* (O<=Z<=L) нулевые на
чальные значения.
2. For все пикселы *P* изображения do:
   Begin.
3. Увеличение *H(f(P))* на 1.
   End.
4. Конец алгоритма.

Такую гистограмму можно использовать для улучшения качества или рационального кодирования изображения. Может оказаться, что для многих значений *Z H(Z)=0;* это значит, что допустимые уровни квантования используются неэффективно. Целесообразно в таком случае изменить их задание таким образом, чтобы динамический диапазон изображения увеличился. На рис. 3 3 представлены фотография, снятая с недодержкой, и соответствующая гистограмма. Как видно, число пикселов, значения которых лежат в пределах 180—255, невелико. Можно было бы попытаться использовать весь диапазон с помощью изменения масштаба, одно это не всегда удается, а в данном случае явно неосуществимо. Более удачный способ использования допустимого диапазона воспроизведения полностью состоит в изменении задания значений уровней серого тона таким образом, чтобы соответствующая гистограмма оказывалась как можно более ровной.

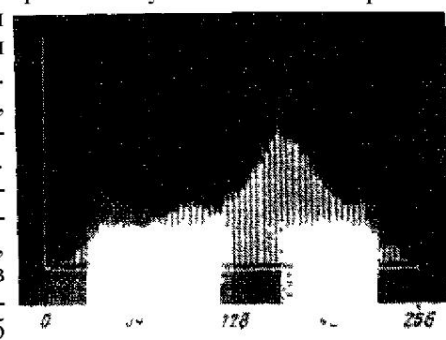


Рис. 3 2. Гистограмма уровней серого тона для изображения, приведенного на рис. 2 3

Пусть *A* — площадь изображения, а *N* — число допустимых Уровней яркости. Для получения абсолютно ровной гистограммы необходимо, чтобы на каждый уровень приходилось по *A/N* пикселов. Если значения яркости некоторого уровня *Z* в *k* раз больше среднего, то этот уровень следует отобразить в *k* отдельных Уровней, скажем, Z_1; ..., Z_k. Следовательно, необходимо ввести ка-

кое-то правило для выполнения подобного многозначного отображения. Общего метода, который хорошо работал бы во всех прикладных задачах, не существует, однако рассмотрим следующие три возможности.

Правило 1. Уровень Z всегда отображается на средний уровень $(Z_1+Z_k)/2$. (Получаемая гистограмма не является ровной, однако уровни яркости на ней разнесены.)

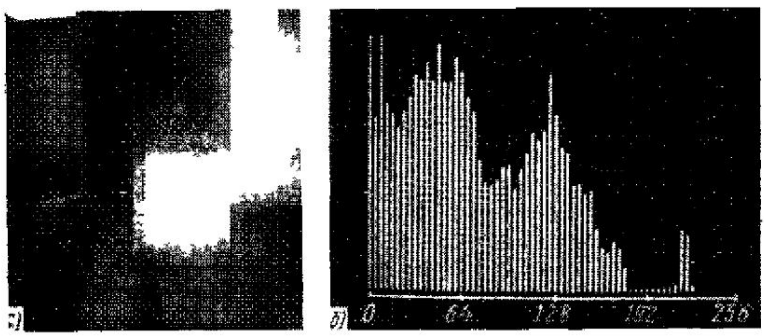


Рис. 3 3 Изображение и соответствующая гистограмма, показывающая, что не все уровни яркости используются эффективно

Правило 2. Случайным образом выбирается один из уровней, входящих в интервал $[Z_1\ Z_k]$. (Эта операция может привести к потере контрастности, если исходная гистограмма содержала два далеко разнесенных пика.)

Правило 3, Изучается окрестность пиксела и ей присваивается то значение уровня из интервала $[Z_1, Z_2]$, которое ближе всего к среднему значению яркости выбранной окрестности. (Эта операция может вызвать размывание краев и требует объема вычислений, существенно большего, чем при использовании предыдущих двух правил.)

Правило 1 является сугубо эвристическим. Оно не обеспечивает получение по-настоящему выровненных гистограмм, однако позволяет полностью использовать динамический диапазон ценой меньших, чем при использовании других правил, затрат. (При решении одной конкретной задачи правило 2 вследствие необходимости обращения к рандомизации потребовало почти четырехкратных затрат времени по сравнению с первым.) Побудительным мотивом для использования правила 2 служит стремление исключить систематическую ошибку: если выбор является произвольным, то реализовать его необходимо некоторым случайным образом. Правило 3 отражает попытку обеспечить определенную согласованность уровней яркости отдельных пикселов. Предполагается, что чем ближе два пиксела друг к другу, тем вероятнее близость их значений яркости. (Более подробное обсуждение этой темы см. в разд. 3.4.)

Алгоритм 3.2. Выравнивание гистограммы

Обозначения. H — массив гистограммы, $H_{сов}$ — некоторое совокупное значение гистограммы, Z обозначает исходные уровни яркости, *а* R — новые. Каждый уровень Z отображается на интервал *[левый (Z), правый (Z)]*.

0. Считывание изображения, оценивание его гистограммы и занесение в массив Я. Пусть $H_{ср}$ — среднее значение.
1. Установка $R = 0$ *и* $H_{сов} = 0$.
2. For Z = 0 до L do.
   Begin.
4. Установка *левый (Z)*=R и прибавление H(Z) к $H_{сов}$
5. While $H_{сов}$ больше, чем $H_{ср}$, do:
   Begin.
6. Вычитание $H_{ср}$ из $H_{сов}$ и увеличение R.
   End.
7. Установка *правый (Z)*=R и задание значения *новый (Z)* в соответствии с используемым правилом. По правилу 1 *новый* (Z) приравнивается среднему от значений *левый (Z)* и *правый* (Z). По правилу 2 назначение *новый* (Z) приравнивается разности *правый* (Z) — *левый* (Z). (По правилу 3 значение *новый* (Z) остается определенным.)
   End.
8. For всех пикселов P изображения do:
   Begin.
9. If *левый (f(P))* равен *правый (f(P))*, then новое значение пиксела P приравнивается значению *левый (f(P))*.
10. Else:
   Если используется правило 1, значение пиксела приравнивается новый *(f(P))*.
   Если используется правило 2, в интервале [0, *новый (f(P))]* случайным образом выбирается точка, ее значение
   прибавляется к значению *левый (f(P))* и полученный результат рассматривается как новое значение яркости пиксела P.
   Если используется правило 3, вычисляется среднее значение по окрестности пиксела P. Если оно больше значения *правый* (Z), последнее рассматривается в качестве нового значения. Если указанное среднее меньше значения *левый* (Z), в качестве нового значения рассматривается последнее. В остальных случаях в качестве нового значения рассматривается среднее. End.
11. Конец алгоритма.

Алгоритм 3.2 позволяет выравнивать гистограмму в соответствии с любым из перечисленных выше правил. Шаг 0 обеспечивает подготовку гистограммы в соответствии с алгоритмом 3.1. Шаги 1—7 обеспечивают выравнивание гистограммы, осуществляе-

мое при помощи преобразования исходных уровней яркости в новые Шаги 8—10 обеспечивают воспроизведение нового изображения На рис 34 представлены результаты применения этого алгоритма к изображению, приведенному на рис 3 3

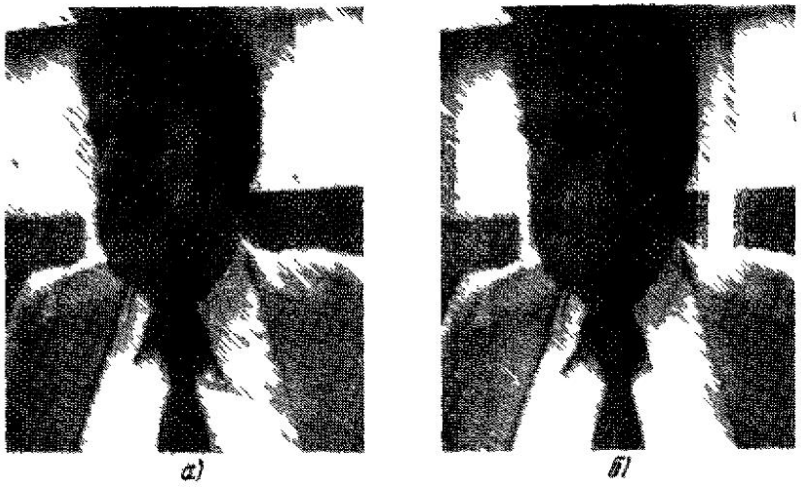


Рис 34 Результаты выравнивания гистограммы изображения представленного на рис 3 3
а — изображение полученное с использованием правила 1 б —правила 2

Рассматриваемый вид обработки нельзя применять к изображениям без разбора, поскольку выравнивание гистограммы часто может вызывать ухудшение его качества Так, на рис 3 5 представлены изображение и его гистограмма, а на рис 36 — соответствующие результаты выравнивания вследствие растяже-

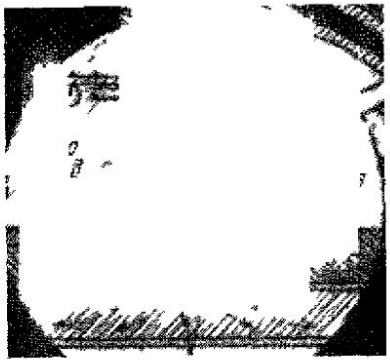


Рис 3 5 Исходное изображение и его гистограмма

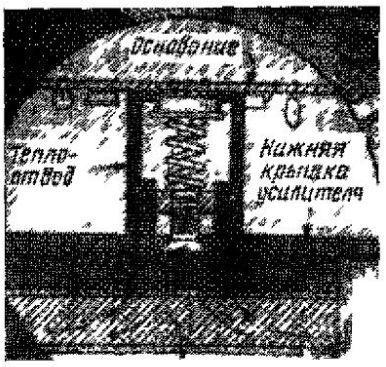


Рис 3 6 Ухудшение качества изображения вызванное выравниванием его гистограммы

ния бимодальной гистограммы выравненное изображение выглядит «загрязненным»

Выравнивание гистограммы можно использовать и для получения изображения с более грубым характером квантования. Например, встречаются ситуации, когда при дискретизации изображения можно использовать значительное число уровней серого тона (скажем, 256), но изображение воспроизводится на устройстве, допускающем передачу лишь восьми уровней При обращении к алгоритму 3 2 L можно положить равным новому максимальному значению яркости (в данном случае 7) Это же значение следует использовать при определении среднего значения гистограммы $H_{ср}$ Отметим, что в данном случае мало вероятно отображение некоторого отдельного уровня серого тона в несколько уровней, и, следовательно, шаги алгоритма 7 и 10 можно упростить На рис 3 7 представлены результаты применения подобного метода квантования к изображению, помещенному на



а) *б)*

Рис 3 7 Оптимальное квантование изображений, достигнутое с помощью выравнивания гистограммы

а- использовано четыре уровня и *б* — восемь уровней
Качество изображений явно лучше качества изображений полученных с помощью равномерного квантования (см рис 2 9 и 2 10)

рис 23. Сравнение рис 3 . 7 с рис 2.9 и 2.10 ясно показывает, что учет лишь самых старших разрядов не является оптимальным решением Так можно представить себе прикладные задачи, в которых стоимость препроцессора, выравнивающего гистограмму, оказывается меньше затрат на дополнительную память обновления изображения, необходимую для воспроизведения изображения с большим числом разрядов Если ограничения связаны не со стоимостью памяти, а с оптическими характеристиками устройства отображения, то для выравнивания гистограммы можно воспользоваться функцией типа *color (u, v),* описанной в разд. 17 (см табл. 1 1)

Родственная проблема возникает и при использовании *псевдоцвета.* В некоторых прикладных задачах для обнаружения на

изображении важной «информации» требуется различать близкие уровни серого тона. Выравнивание гистограммы можно применять для расширения диапазона яркостей, встречающихся в изучаемой области изображения. С другой стороны, соответствующие уровни можно представлять не с помощью шкалы серого тона, а *цветом.* Если количество цветов, которые можно использовать, окажется небольшим, то для решения этой задачи нам также придется обратиться к выравниванию гистограммы.

Пример 3.3. При решении многих задач медицинской диагностики и контроля качества в промышленности требуется отыскивать на радиографическом изображении темные пятна. Поскольку форма этих пятен имеет существенное значение для диагностики, они должны тщательно отделяться от фона. При осмотре исходного изображения такие «патологические» пятна можно перепутать с «тенями». Одной из разновидностей улучшения качества изображения является повышение разрешающей способности по яркости в более темных участках изображения. Если диапазон яркостей исходного изображения составляет [0,255], то подобное улучшение качества может быть осуществлено посредством отображения диапазона [0,31] в диапазон [0,254] (с помощью алгоритма 3.2, использующего правило 1) и диапазона [32,255] в значение 255. В результате значения яркости должны отличаться друг от друга на 10% или более. Другой вариант решения заключается в том, что учитывается лишь часть гистограммы, относящаяся к диапазону [0,31], — она отображается в диапазон, состоящий из восьми значений, каждому из которых ставится в соответствие с помощью команды *color (u,* и) (см. табл. 1.1) один из восьми цветов.

3.3. МАТРИЦЫ СОВМЕСТНОЙ ВСТРЕЧАЕМОСТИ УРОВНЕЙ СЕРОГО ТОНА

Матрица совместной встречаемости уровней серого тона представляет собой оценку плотности распределения вероятностей второго порядка $p_2(P, Q, Z, У)$, полученную по одному изображению в предположении, что плотность вероятности p_2 зависит лишь от взаимного расположения P и Q. Обозначим матрицу через $C_r(Z, Y)$, где r — отношение, в котором находятся P и Q. Поскольку число таких матриц может быть очень большим, необходимы дальнейшие упрощения. Во-первых, можно ограничиться учетом лишь непосредственных соседей соответствующего пиксела: например, пиксела, находящегося над ним или справа от него. Во-вторых, можно полностью пренебречь ориентацией, усредняя матрицы, сформированные для различных ориентации. Алгоритм 3.3, представляющий модификацию алгоритма 3.1, обеспечивает получение таких матриц. Если требуется усреднение матриц, то установка начальных нулевых значений должна выполняться за пределами первого цикла, а окончательный результат следует делить на число использованных отношений.

Алгоритм 3.3. Построение матриц совместной встречаемости уровней серого тона

1. For всех отношений r, заданных на парах пикселов, do:
   Begin.
2. Установка нулевых значений в массиве C_r(Z,У) (0<=Z<=L, 0<=Y<=L).

3. For всех пикселов P изображения **do:**
Begin.
4. Пусть Q — пиксел, соединений с пикселом P по отношению r. Увеличение $C_r(f(P), f(Q))$ на 1.
End.
End.

5. Конец алгоритма.

Может оказаться, что в случаях, когда пиксел P находится вблизи края изображения, пиксел Q, определяемый отношением r, не существует. Тогда, вместо значения $f(Q)$ можно подставлять ноль или некоторое другое специально оговоренное значение. В дальнейшем всегда будем считать пикселы P и Q смежными. Кроме того, в примерах, приводимых в данном разделе, будут рассматриваться только два отношения: пиксел Q расположен справа от пиксела P (отношение 1) и пиксел Q расположен над пикселом P (отношение 2). Многие свойства матрицы C_r явно определяются способом ее формирования. Во-первых, диагональные элементы матрицы приблизительно равны площадям соответствующих областей изображения — элемент $Cr(k, k)$ равен площади областей изображения, содержащих пикселы, значения яркости которых равны k. Во-вторых, элементы матрицы, находящиеся вне диагонали, приблизительно равны длинам границ, разделяющих соответствующие области изображения, элемент $C_r(k, j)$ равен длине границы, разделяющей области, которые образованы пикселами с яркостями k и j. Если изображение малоконтрастное, то элементы, удаленные от диагонали, равны нулю или имеют очень малое значение; если изображение высококонтрастное, то ситуация прямо противоположная. Основная трудность, возникающая при использовании матриц совместной встречаемости, связана с их большими размерами (LxL). Для преодоления этой трудности можно при оценивании матрицы перейти к более грубому квантованию. Добиться этого можно с помощью описанного в предыдущем разделе метода, основанного на выравнивании гистограммы.

Пример 3.4. Пусть изображение, использованное в примере 3.2, имеет размеры $N*N$, является двухуровневым и ширина каждой его полосы составляет пять пикселов. В таком случае матрицы совместной встречаемости для двух указанных отношений имеют следующий вид (приближенно):

$$C_1 = \begin{bmatrix} 4N/10 & N/10 \\ N/10 & 4N/10 \end{bmatrix}, \qquad (3.1а)$$

$$C_2 = \begin{bmatrix} N/2 & 0 \\ 0 & N/2 \end{bmatrix}. \qquad (3.1б)$$

Отметим, что эти матрицы не меняются при изменении изображения, если число полос и их средняя ширина остаются постоянными.

На рис. 3.8 представлены матрицы совместной встречаемости, соответствующие отдельным частям изображения, приведенного на рис. 2.3. Для уменьшения размеров матрицы использовано восемь уровней квантования. Значения этих уровней были определены с помощью выравнивания гистограммы по пол-

ному изображению и поэтому результат оказался таким же, какой получен для изображения, приведенного на рис. 3.7,б. Матрица а) соответствует квадрату 128X128, «вырезанному» на портьере, матрица б) — квадрату 90X90 на волосах и матрица в) — квадрату 80X80 на блузке. Каждая матрица представляет собой результат усреднения по четырем отношениям: выше, ниже,

	1	2	3	4	5	6	7	8
1	0	0	0	0	0	0	0	0
2	0	0	0	0	0	0	0	0
3	0	0	353	444	17	0	0	0
4	0	0	444	5791	1902	595	17	0
5	0	0	17	1902	2452	2077	138	0
6	0	0	0	595	2077	5970	1812	5
7	0	0	0	17	138	1812	3676	51
8	0	0	0	0	0	5	51	17

а)

	1	2	3	4	5	6	7	8
1	483	422	9	8	0	0	0	0
2	422	7350	1450	71	0	0	0	0
3	9	1450	3068	413	1	0	0	0
4	8	71	413	267	4	0	0	0
5	0	0	1	4	0	0	0	0
6	0	0	0	0	0	0	0	0
7	0	0	0	0	0	0	0	0
8	0	0	0	0	0	0	0	0

б)

	1	2	3	4	5	6	7	8
1	650	127	31	32	21	29	28	9.
2	127	535	124	55	18	21	41	48
3	31	124	435	142	33	39	51	110
4	32	55	142	667	113	88	85	172
5	21	18	33	113	253	179	55	108
6	29	21	39	88	179	317	111	201
7	28	41	51	85	55	111	109	338
8	91	48	110	172	108	201	338	4589

в)

Рис. 3.8. Примеры матриц совместной встречаемости уровней серого тона

слева, справа. Совершенно очевидно, что эти матрицы содержат информацию, характеризующую *текстуру,* а также среднюю яркость изображения. Несмотря на достаточную яркость того, что такое текстура, формально определить этот объект весьма непросто. Можно считать, что два изображения, порождающие один набор матриц совместной встречаемости, имеют одинаковую текстуру (см. разд. 3.6).

3.4. ЛИНЕЙНАЯ ФИЛЬТРАЦИЯ ИЗОБРАЖЕНИЙ

Значительную часть обработки изображений можно выполнить, не повторяя для каждого нового изображения статистический анализ, описанный в предыдущем разделе. Вполне достаточным оказывается ограниченный объем априорных сведений. Допустим, нам известна матрица совместной встречаемости для «идеального» изображения и требуется улучшить качество зашумленного варианта этого изображения. Если наибольшие элементы матрицы расположены на главной диагонали или вблизи нее, то это означает, что большая часть пикселов имеет тот же цвет, что и соседние пикселы. Если мы хотим выровнять гистограмму такого изображения, то, как показано в разд. 3.2, целесообразно воспользоваться правилом 3. Если требуется устранить шум, то замена значения каждого пиксела зашумленного изображения некоторой взвешенной суммой значений соседних пикселов приведет к уменьшению изменчивости значений смежных пикселов, и мы получим изображение, более близкое к оригиналу (см. пример 3.5). Таким образом мы приходим к соотношению, характеризующему связь исходного изображения $f(x, y)$ и изображения $g(x, y)$, *подвергнутого фильтрации:*

$$g(x, y) = \sum_{i=-M}^{M} \sum_{j=-M}^{M} h(x, y, i, j) f(x+i, y+j). \tag{3.2}$$

Процесс, реализующий эту операцию, называют *линейным фильтром,* в частности *фильтром скользящего среднего,* поскольку при его использовании значение каждого пиксела заменяется разновидностью среднего от значений соседних с ним элементов. Если весовая функция h в пределах изображения не изменяется и не зависит от координат x, y, то уравнение (3.2) можно переписать в следующем виде:

$$g(x, y) = \sum_{i=-M}^{M} \sum_{j=-M}^{M} h(i, j) f(x+i, y+j). \tag{3.3}$$

Этот процесс называют *пространственно-инвариантным фильтром.* Эти фильтры широко применяются при обработке временных сигналов, однако целесообразность их использования в обработке изображений не очевидна. Уравнение (3.3) упрощается при записи его через фурье-преобразование. Можно показать (см. разд.

$$G(u, v) = H(u, v) F(u, v). \tag{3.4}$$

Результат применения такого фильтра состоит в подавлении одних частот и усилении других в зависимости от $H(u, v)$.

Пример 3.5. Если требуется очистить изображение от высокочастотного шума, то для осуществления такого сглаживания можно воспользоваться следующей разновидностью функции $h(i, j)$:

$$h(0, 0) = 1/4, \tag{3.5а}$$

$$h(1,\ 0) = h(0,\ 1) = h(0,\ -1) = h(-1,\ 0) = 1{,}8, \qquad (3\,5б)$$

$$h(1,\ 1) = h(1,\ -1) = h(-1,\ 1) = h(-1,\ -1) = 1/16. \qquad (3\,5в)$$

Результат применения фильтра можно оценить сравнив разности значении соседних пикселов до и после фильтрации В частности простейшие выкладки показывают что

$$g(x,\ y) - g(x+1,\ y) = \frac{1}{8}[f(x,\ y) - f(x+1,\ y)] +$$
$$+ \frac{1}{8}[f(x-1,\ y) - f(x+1,\ y)] +$$
$$+ \frac{1}{16}[f(x\ \ y-1) + f(x,\ y+1) - f(x+1,\ y-1) -$$
$$- f(x+1,\ y+1) + f(x-1,\ \ y-1) + f(x-1\ \ y+1) -$$
$$- f(x+2,\ \ y-1) - f(x+2,\ \ y+1)] \qquad (3\,6)$$

Если обозначить через D_f максимум абсолютной разности значений смежных пикселов исходного изображения $f(x, y)$ и через D_g соответствующую разность для пикселов изображения, прошедшего фильтрацию, то из уравнения (36) следует

$$D_g \leqslant \frac{1}{8} D_f + \frac{3}{8} Df + \frac{1}{16} 4\ 2\ D_f = D_f, \qquad (3\,7)$$

т.е. очевидно, что эта разность расти не может Равенство имеет место только в случаях, когда максимальная разность значений n пикселов в n раз больше максимальной разности значений для пары пиксетов т.е. когда $f(x\ y)$ есть некоторая линейная функция от своих аргументов В противном случае указанная разность будет уменьшаться и области изображения будут принимать более однородный характер Поскольку обычно применение такого простейшего фильтра оказывается недостаточным для устранения шума приходится прибегать к использованию фильтра высшего порядка Одна из возможных реализаций такого фильтра заключается в выборе простого фильтра и многократном применении его к изображению На рис 39 и 310 приведены результаты фильтрации на рис 3 10 а представлено исходное изображение полученное из изображения, приведенного на рис 3 9, при помощи наложения на последнее *гауссовского белого шума,* на рис 3.10 б представлены результаты восьмикратного применения фильтра заданного уравнением (3 5) Не трудно видеть что этот процесс приводит не только к удалению высокочастотного шума, но вызывает и размывание краев изображения



Рис 3 9 Исходное изображение использованное для

Действительно, если применить уравнение (3.6) к четко очерченному краю изображения

$I(x\ y) = f$ при $x<=X$ $f(x\ y)=0$ при $x>X$,

то оказывается, что $g(X, y) - g(X+1, y) = 1/8+1/8+4/16=1/2$

Другими словами, разность значений смежных пикселов уменьшилась вдвое и это, естественно привело к понижению контрастности изображения

Результаты применения данного фильтра можно также оценивать, рассматривая фурье-преобразование функции $h(i, j)$ Введем обозначение $E(x) = \exp(i\frac{2\pi}{N}x)$

Воспользуемся уравнением (34)

$$H(u\ \ v) = \frac{1}{4} + \frac{1}{8}[E(-v) + E(v) + E(-u) + E(u)] +$$

$$+ \frac{1}{16}[E(-(u+v)) + E(-(u-v)) + E(u-v) + E(u+v)] \qquad (3\,3)$$

Это выражение можно упростить, воспользовавшись тождеством

$$\cos x = \frac{1}{2}[e^{i x} + e^{-i x}]$$

и выполнив затем простейшие тригонометрические преобразования, в результате выражение (3 8) принимает следующий вид

$$H(u, v) = \frac{1}{4}\left(1 + \cos\frac{2\pi}{N}v + \cos\frac{2\pi}{N}u + \cos\frac{2\pi}{N}v\cos\frac{2\pi}{N}u\right) \qquad (3\,9)$$

Отметим, что H=0, если u или v равны $N/2$, и что, кроме того,

$H(0, 0)$=1, $H(0, N/4) = H(N/4, 0)$ - 1,2, $H(N/4, N/4)$ = 1/4,

Таким образом, высокие частоты подавляются



Рис 3 10. Изображение с высоким уровнем шумов, полученное путем наложения шума на изображение рис 39 (а), изображение, полученное в результате применения процедуры линейной фильтрации к изображению, приведенному на рис 3.10,а *(б)*

Наличие отмеченной связи между удалением шума и размыванием краев изображения означает, что при обработке изображений линейными пространственно инвариантными фильтрами следует пользоваться осмотрительно К сожалению, обширное и успешное применение подобных фильтров при работе с временными функциями стимулировало их использование в обработке изображений без учета присущих им ограничений

Аналогичная проблема возникает в связи с *фильтрами верхних частот* Их использование позволяет получать изображения с более резкими краями однако одновременно приводит к усилению высокочастотного шума

Пример 3.6. Простейший фильтр верхних частот определяется следующим образом:

$$h(0,\ 0) = b, \tag{3.10а}$$

$$h(1,\ 0) = h(-1,\ 0) = h(0,1) = h(1,\ 0) = -a. \tag{3.10б}$$

Его фурье-образ имеет вид

$$H(u,\ v) = b - 2a\left(\cos\frac{2\pi}{N}v + \cos\frac{2\pi}{N}u\right). \tag{3.11}$$

Для H максимум составляет $b + 4a$ и достигается при $u=v=N/2$, а минимум $H(0, 0)=b-4a$.

3.5. НЕЛИНЕЙНАЯ ФИЛЬТРАЦИЯ ИЗОБРАЖЕНИЙ

Фильтры, применение которых обеспечивает именно очищение от шума участков, расположенных внутри определенных областей изображения, не вызывая при этом размывания его краев, значительно сложнее фильтров, определяемых уравнением (3.3). Такие фильтры должны до реализации сглаживающей функции попытаться обнаружить края изображения. Эту задачу нельзя назвать легкой, поскольку непросто обнаружить край на зашумленном изображении. Рассмотрим несколько наиболее распространенных подходов к ее решению.

3.5.1. НАПРАВЛЕННЫЕ ФИЛЬТРЫ

Один из методов предполагает использование линейного фильтра, симметричного относительно некоторой оси, а не точки. В этом случае при обработке каждого пиксела предпринимается попытка оценить направление границы изображения (если она имеется) и фильтр не допускает усреднения точек в направлении, поперечном границе. Эту идею можно довольно просто реализовать, введя две функции угла φ , образуемого краем изображения с осями координат:

φ°	0	45	90	135
$c(\varphi)$	1	1	0	−1
$s(\varphi)$	0	1	1	1

Тогда функция, реализуемая фильтром, $h(i, j, \varphi)$ определяется как

$$h(0, 0, \varphi) = 0{,}5\ ; \quad h(C(\Phi), s(\varphi), \varphi) = h(-c(\varphi), -s(\varphi), \varphi) = 0{,}25;$$

при всех остальных значениях аргументов i и j эта функция равна нулю. В каждом пикселе значение выражения

$$V(\varphi) = [f(x, y) - f(x + c(\varphi), y + s(\varphi))]^2 + [f(x, y) - f(x - c(\varphi), y - s(\varphi))]^2$$

вычисляется при φ=0, 45, 90 и 135°. Затем фильтр применяется соответственно тому значению угла φ, при котором значение $V(\varphi)$ является минимальным. Результат применения такого фильтра представлен на рис. 3.11.

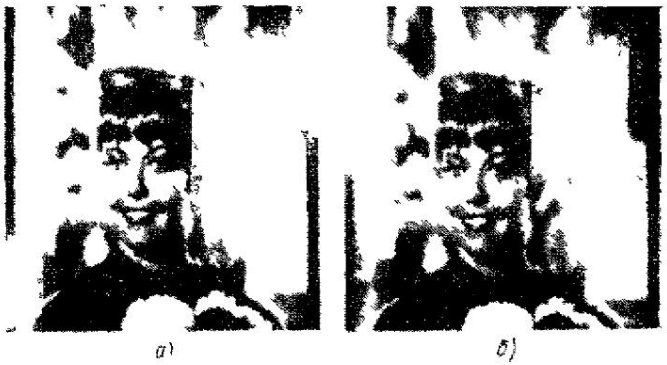


Рис. 3.11. Результаты применения направленного фильтра нижних частот к изображению, приведенному на рис. 3 10,*а*:
а — изображение, полученное после однократного и б — трехкратного применения фильтра

3.5.2. СОСТАВНЫЕ ФИЛЬТРЫ

Составной фильтр представляет собой развитие идеи направленного линейного фильтра. Этот вид фильтрации предусматривает в первую очередь применение к изображению фильтра нижних частот и вычисление градиента исходного изображения $f(x, y)$. Эта процедура позволяет получить некоторое представление о расположении краев изображения. Затем к изображению полученному в результате первого шага фильтрации, снова применяется фильтр, но его коэффициенты уже зависят от местоположения обрабатываемого участка на изображении и поэтому края изображения размываться не будут. Если исходное изображение не сохранилось, то к изображению, подвергнутому фильтрации, можно применить фильтр верхних частот, коэффициентам которого при обработке участков изображения с большим значением градиента присваиваются большие значения, а при обработке остальной части изображения — малые. Эта процедура должна обеспечить восстановление значений элементов исходного изображения на участках с высоким значением градиента. Если же исходное изображение имеется, то целесообразно воспользоваться фильтром

нижних частот. Известны различные варианты усовершенствования этого подхода; мы, кроме того, предлагаем читателю продумать собственные варианты такой модификации (см. разд. 3.6)[1]

3.5.3. ФИЛЬТРЫ ПРИБЛИЖЕНИЯ ФУНКЦИЙ

Принцип действия фильтров этого типа основывается на замене исходного изображения *f(x, y)* локальной оценкой одной из ее статистических характеристик. Для определения соответствующих статистических характеристик необходимо разбить изображение на участки с некоторыми фиксированными размерами. В простейшем варианте этого метода на каждом из выделенных участков изображения оцениваются среднее значение и дисперсия функции *f(x, y)*. Если значение дисперсии оказывается ниже заданного порога, то значения *f(x, y)* заменяются ее средним. В противном случае соответствующий участок разбивается на более мелкие и процедура повторяется. Таким способом обеспечивается выделение краев. Вместо оценивания среднего значения можно обратиться к приближению *f(x, y)* некоторой гладкой функцией и переходить к разбиению участка на более мелкие, если ошибка приближения оказывается чрезмерно большой. (Соответствующие структуры данных обсуждаются в гл. 6, а методы приближения — в гл. 12.) Этот подход также тесно связан с сегментацией изображения, рассматриваемой в следующей главе.

3.6. БИБЛИОГРАФИЧЕСКИЕ ЗАМЕЧАНИЯ

Проблемы фильтрации и обнаружения краев изображений обсуждаются в работах [2.1, 2.3, 2.9, 2.11], упоминавшихся в предыдущей главе.

Вывод уравнения (3.4) можно найти в самых элементарных учебниках по обработке сигналов (например, [3.6 или 3.8]). Более подробную информацию о нелинейных фильтрах следует искать в периодике; по поводу направленных фильтров можно обратиться к работе [3.5], а по поводу составных фильтров — к работе [3.1]. Гистограммы рассматриваются практически во всех книгах, посвященных обработке изображений и распознаванию образов [2.9, 2.11, 3.7 и т. д.]. Связь между матрицами совместной встречаемости уровней серого тона и текстурой впервые была изучена Хараликом; подробное обсуждение феномена текстуры имеется в его обзорной статье [3.2]. Джулес изучил различение человеком образов, составленных из случайно расположеных точек, на основе их статистик высших порядков [3.3, 3.4]. Его работы являются еще

[1] Фильтры такого типа можно было бы называть двойными, или двукратными (two-pass), однако нам кажется более предпочтительным термин «составной» (two-part), поскольку на первом и втором шагах фильтрации используются разные фильтры. Таким образом нам удается разграничить эти нелинейные и линейные фильтры, многократное применение которых является просто средством реализации фильтров высших порядков.

одним подтверждением существования связи между статистиками второго порядка и текстурой.

Теоретическая работа в области обработки изображений осложняется наличием одной серьезной проблемы: отсутствием математического критерия, отражающего субъективные различия восприятия изображений. Допустим, имеется изображение $f(x, y)$ и мы хотели бы установить, какое из двух изображений $f_1(x, y)$ и $f_2(x, y)$ является более точным приближением заданного изображения. Ни одна из известных мер различия двух функций не позволяет получить результаты, согласующиеся с оценкой человека, наблюдающего соответствующие изображения. Обсуждение этой темы можно найти в работе [2.12].

3.7. ЗАДАЧИ

3.1. Включите алгоритм выравнивания гистограммы — алгоритм 3.2 — в программу вывода тоновых изображений посредством печати с наложением, составленную в соответствии с условием задачи 1.6.

3.2. Напишите программу определения матрицы совместной встречаемости уровней серого тона.

3.3. *Доминирование диагонали* в матрице с неотрицательными элементами можно оценить с помощью отношения суммы диагональных элементов к сумме элементов, расположенных вне диагонали. Какого рода информацию относительно вида изображения несет коэффициент доминирования диагонали матрицы совместной встречаемости уровней серого тона D? Может ли D быть равным нулю? — бесконечности? Воспользуйтесь программой из задачи 3.2 для определения коэффициента D на отдельных участках различных изображений.

3.4. Это задание предполагает, что Вы работаете над созданием программы-редактора изображений (см. задачи 1.5 и 2.5). Вы можете теперь расширить его возможности, включив в него процедуру из задач 3.1 и 3.2 Основная проблема, которую следует при этом разрешить, заключается в том, где и каким образом будет воспроизводиться гистограмма или матрица совместной встречаемости уровней серого тона.

Указание. Имеется две возможности: либо отвести часть экрана для воспроизведения статистических характеристик заданной области, либо отвести отдельные части памяти обновления изображения для хранения этих статистических характеристик и затем использовать два «цветных» отображения (см. команду 13 из табл. 1.1) для выполнения необходимых переходов. Целесообразно выбирать масштаб значений элементов матрицы совместной встречаемости уровней серого тона таким образом, чтобы эту матрицу можно было воспроизводить в виде некоторого изображения.

3.5. Синтезируйте фильтр нижних частот для изображений, о которых известно, что наибольшая часть их краев направлена либо по вертикали, либо по горизонтали.

3.6. Для реализации процедуры, определяемой уравнением (3.3), необходимо располагать двумя «экземплярами» изображения — входным и выходным. Допустим, что вместо этого мы совершаем обход изображения, заменяя значение каждого пиксела взвешенной суммой, определяемой правой частью уравнения (3.3). Это значит, что в дальнейших вычислениях будет использоваться новое значение. Изучите результат применения такого фильтра.

3.7. Рассмотрим следующий нелинейный фильтр: для каждого пиксела вычисляется разность D_v значений двух пикселов, являющихся его соседями сверху и снизу, а также разность D_h значений двух пикселов, являющихся его соседями справа и слева. Если $D_v > D_h$, то значение соответствующего пиксела заменяется взвешенным средним значением этого пиксела и двух смежных с ним по горизонтали пикселов; в противном случае значение пиксела заменяется взвешенным средним значением этого пиксела и двух смежных с ним

по вертикали пикселов (Замена значений осуществляется в соответствий с уравнением (3 3) а не так как это делалось в задаче 3 6) Изучите результат применения такого фильтра при различных комбинациях весов Изучите их также при многократном применении фильтра к некоторому изображению

3 8 Придумайте устройство, которое могло бы помочь фотографу-любителю, обладающему деньгами в большей мере, чем талантом Оно должно представлять собой некий черный ящик, принимающий фотографии, снятые с передержкой или недодержкой, и выдающий экземпляры с должным уровнем освещенности Оно должно также улучшать качество плохо сфокусированных изображений Предусмотрите, чтобы фотографу не надо было указывать, какого рода коррекция изображения необходима Считаете ли Вы что можно предусмотреть исправление кадров в которых объект двигался или кадров, снятых перемещающимся фотоаппаратом?

Глава 4

СЕГМЕНТАЦИЯ

4.1. ВВЕДЕНИЕ

Сегментация позволяет выделить участки изображения, кажущиеся наблюдателю однородными; эта операция обеспечивает разбиение изображения на области одинакового вида Сравнительно нетрудно определить однородность в категориях уровней серого тона или цвета Значительно сложнее определить, что именно мы имеем в виду, говоря об «однородной текстуре»

В принципе сегментацию можно осуществлять двумя способами Можно считать, что характеристики областей нам известны заранее, но можно определять эти характеристики и в процессе обработки В качестве простейшего примера рассмотрим сегментацию по цвету изображения, в представлении каждого пиксела которого по 2 бита отведено на каждый из трех основных цветов (красный, зеленый и синий) Чтобы выделить области изображения красного цвета, необходимо у каждого пиксела проверить равенство единице соответствующих битов и равенство нулю остальных битов, отведенных для кодирования цвета [1] С другой стороны, возникает более трудная задача, если требуется разделить изображение на две группы областей, в каждую из которых входят области одного цвета, причем сами цвета заранее не заданы В последнем случае приходится либо изучать статистические характеристики изображений и отыскивать моды распределения, либо проверять отдельные группы пикселов, выясняя, образуют ли они однородную область Поскольку обсуждение проблемы сегментации во всей ее полноте вывело бы нас за пределы задач данной книги, ограничимся обсуждением лишь нескольких прос-

[1] Предложенный метод решения примитивен, поскольку он позволяет выделять лишь ярко красные области Более практичный метод должен предусматривать смягчение указанных условий

тейших схем В разд. 4.2 рассматриваются методы разделения по порогу, разд. 4.3 посвящен простым методам обнаружения границ областей, а разд. 4.4 — некоторым методам выделения областей путем наращивания

Анализ сцен представляет собой задачу, связанную с сегментацией Его цель состоит в получении описания изображения на некотором языке Можно разделить эти процедуры и провести анализ после завершения сегментации Однако многие исследователи при выполнении сегментации руководствуются априорной информацией о сцене Подобные методы выходят далеко за пределы обработки сигналов и алгоритмов и потому, естественно, в данной книге не рассматриваются

4.2. РАЗДЕЛЕНИЕ ПО ПОРОГУ

Этот метод предусматривает сопоставление значения яркости каждого пиксела с *пороговым,* в результате чего соответствующий пиксел зачисляется в одну из двух групп в зависимости от того, превышает значение яркости пороговое или нет. Выбор значения порога обычно производится по гистограмме (см разд. 3.2) Допустим, мы работаем с изображениями класса 1, для которых выполняются условия примера 3 1, однако отсутствует информация о форме и местоположении областей *B* и *C* Если изображение действительно состоит из двух областей, в одной из которых преобладают светлые, а в другой темные пикселы, то можно считать, что гистограмма будет иметь два пика, как показано на рис 3 5 В таком случае порог *T* можно выбирать из значений, заключенных между двумя пиками, а область С можно определять как множество таких пикселов, значение яркости которых меньше *T,* На рис 4 1 показаны результаты применения этой процедуры к изображению, приведенному на рис 35 (T = 100) Разделение по порогу представляет собой простейший из возможных методов сегментации (преобразования изображений класса 1 в изображения класса 2) К сожалению, не всегда удается выбирать значение порога заранее, поскольку средний уровень яркости может изменяться, что вызывает необходимость вычисления гистограммы для каждого изображения отдельно

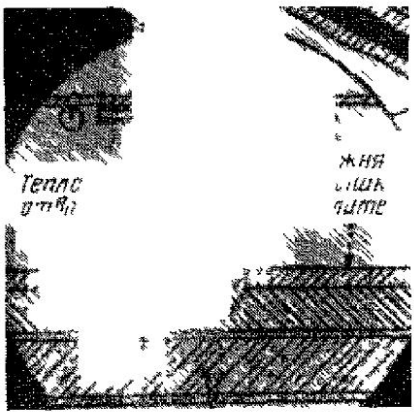


Рис 4 1 Результат сегментации изображения приведенного на рис 35 с использованием процедуры разделения по порогу при T=100

В общем случае автоматическое определение порога — задача нетривиальная Наличие у гистограммы двух пиков — не рядовой

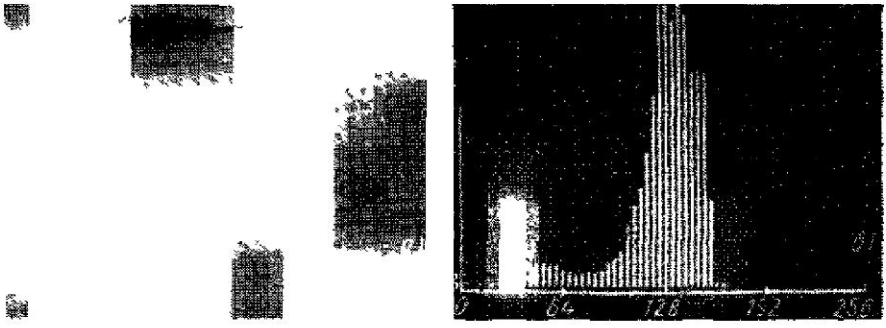


Рис. 4.2. Исходное изображение (256X256 пикселов) (а) и гистограмма уровней серого тона (б)

случай. На рис. 4.2,а приведено изображение, которое кажется вполне пригодным для использования процедуры разделения по порогу, однако его гистограмма (рис. 4.2,б) имеет всего один основной пик, соответствующий фону. Объекты, представленные вторым, слабо выраженным пиком, соответствуют диапазону от примерно 32-го до 64-го уровней серого тона. На рис. 4.3 приведены результаты сегментации с помощью разделения по порогу при $T=100$. Иногда предпочтительнее при построении гистограммы учитывать не все пикселы изображения, а лишь те, которые расположены вблизи границ областей Это легко осуществить, введя в алгоритм 3.1 процедуру сравнения значения пиксела со значениями соседних с ним пикселов. В таком случае $H\ (f(p))$ получает приращение, если разность сопоставляемых значений яркости оказывается больше, чем ожидаемое значение шума. Введение этого ограничения основано на том обстоятельстве, что зрительное восприятие очень чувствительно к изменениям уровней яркости и в меньшей степени — к их абсолютным значениям.



Рис. 4.3. Результат сегментации изображения, приведенного на рис. 4.2,*а*, с использованием процедуры разделения по порогу для T =100

4.3. ОБНАРУЖЕНИЕ ГРАНИЦ

Рассматриваемые схемы предназначены для поиска *границ,* разделяющих отдельные области изображения. Они предусматривают использование оператора *градиента,* после чего для установления того, обнаружена ли граница, применяется процедура разделения по порогу. Затем пикселы, идентифицированные как граничные, должны быть соединены в замкнутые кривые, окружаю-

щие соответствующие области. В этом разделе рассмотрим лишь некоторые простейшие схемы обнаружения границ. Стандартным приемом является учет разности значений двух групп пикселов, как это предусматривает линейный фильтр высоких частот, определяемый уравнением (3.10). Чтобы учитывать различия в направлении границ, необходимо использовать более одного фильтра подобного типа. Если задать некоторую матрицу Я, у которой $H_{ji} = h(i, j)$, то два простейших фильтра имеют следующий вид:

$$H_{\text{гор}} = \begin{bmatrix} -1 & -c & -1 \\ 0 & 0 & 0 \\ 1 & c & 1 \end{bmatrix}, \tag{4.1а}$$

$$H_{\text{верт}} = \begin{bmatrix} 1 & 0 & -1 \\ c & 0 & -c \\ 1 & 0 & -1 \end{bmatrix}. \tag{4.1б}$$

В литературе по обработке изображений часто встречаются работы, в которых описываются фильтры со значениями c, равными 1 или 2 (см. разд. 4.5). Поскольку особенностью фильтров высоких частот является усиление шума, ценность данного класса методов оказывается весьма относительной для обработки зашумленных изображений. На рис. 4.4 и 4.5 представлены результаты применения процедуры обнаружения границ, определяемой уравнениями (4.1) (при c=1), к изображению, приведенному на рис. 3.9. Отметим, что выбор порогового значения, предназначенного для того, чтобы установить, обнаружена ли действительно граница, много проще выбора порогового значения, используемого при сегментации. Еще один пример применения процедуры выделения границ представлен на рис. 4.6. Вследствие более высокой контрастности изображения последние результаты лучше иллюстрируемых на рис. 4.5.



Рис 4 4. Изображение, полученное в результате применения оператора градиента по горизонтали (а), границы, найденные в результате применения оператора градиента по горизонтали и вертикали (б)

Известны и более трудоемкие схемы обнаружения границ, свободные от некоторых недостатков, свойственных описанным нами простейшим схемам, однако они требуют больших вычислительных затрат. Примеры, приведенные на рис. 4.4 - 4.6, могут

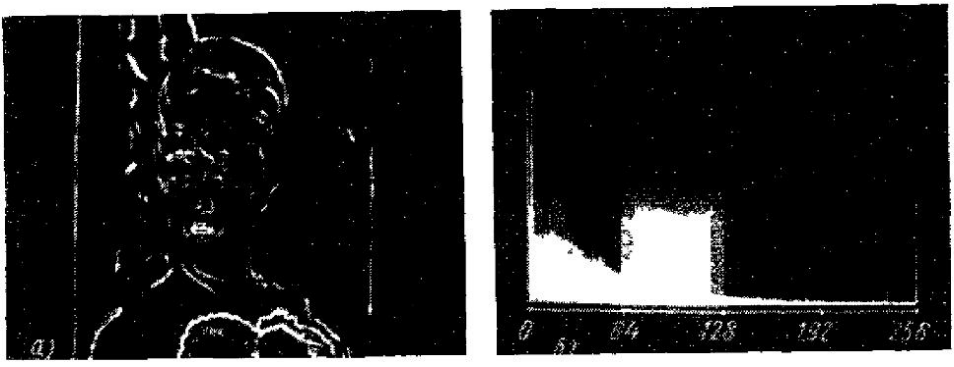


Рис. 45 Гистограмма изображения, приведенного на рис 44,*б* (а), результат сегментации изображения, приведенного на рис 4 4,*б*, с использованием процедуры разделения по порогу для *T=128 (б)*

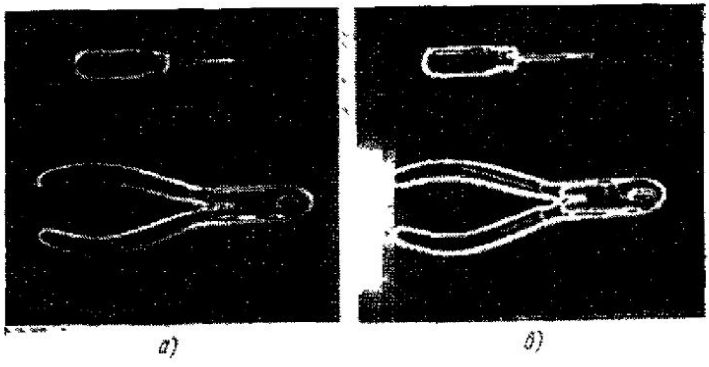


Рис. 46 Границы, выделенные в результате применения оператора градиента по вертикали и горизонтали, на изображении, приведенном на рис. 4 2,*а* (а); результат сегментации изображения, приведенного на рис 4 6,*а*, с использованием метода разделения по порогу для *T* = 64 (б)

создать у читателя впечатление, что изображения, на которых процедуры обнаружения границ позволяют получать хорошие результаты, допускают также применение к ним процедур разделения по порогу, на самом деле это не так. Очевидно, верно, что простые процедуры обнаружения границ годятся лишь для обработки высококонтрастных изображений с низким уровнем высокочастотного шума. Однако не все изображения такого рода можно должным образом сегментировать с помощью процедуры разделения порогу. В частности, *низкочастотный* шум, не препятствующий выделению границ, может лишить эту процедуру всякого смысла.

Простейшим примером такой ситуации служит изображение, характеризующееся градиентом освещенности, — весьма распространенный случай в реальных условиях. Глаз человека чувствителен к контрасту и, следовательно, плавные изменения освещенности может игнорировать. Однако, поскольку то, что выглядит «темным» с одной стороны изображения и «светлым» с другой, имеет одну и ту же яркость, невозможно выбрать порог для разделения «темных» и «светлых» областей. Простая процедура обнаружения границ, с другой стороны, позволяет решить эту задачу вполне удовлетворительно. Такая ситуация может возникать даже в случаях, когда уровень низкочастотного шума не столь высок, поскольку выбор порога для установления границы может оказаться задачей более простой, чем выбор порога для сегментации всего изображения. На рис. 4.7 приведен простой пример сказанного.

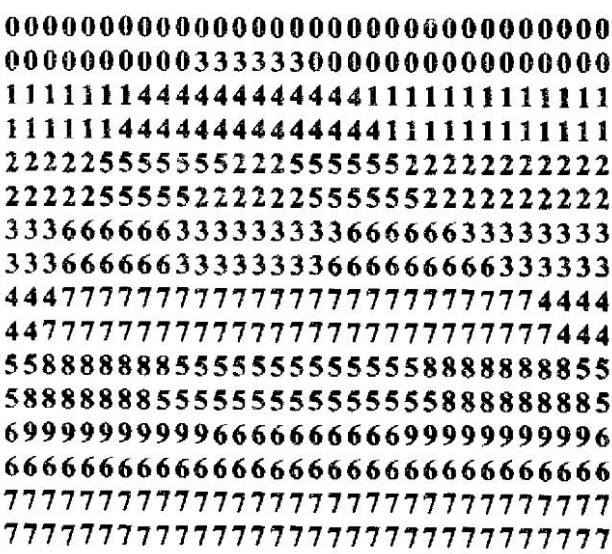


Рис. 4.7. Пример, иллюстрирующий преимущества простой процедуры обнаружения границ перед сегментацией с использованием процедуры разделения по порогу.
Читатели вероятно смогут найти на рисунке очертания символа, а также попробовать разработать систему обнаружения границ для автоматического определения этих очертании (см. задачу 4.1)

4.4. СЕГМЕНТАЦИЯ НА ОСНОВЕ ВЫДЕЛЕНИЯ ОБЛАСТЕЙ ПУТЕМ НАРАЩИВАНИЯ

Если методы обнаружения границ и разделения по порогу основаны на определении разности значений яркости пикселов, то метод выделения областей путем их наращивания предусматривает поиск групп пикселов с близкими значениями яркости. В

простейшем варианте этот метод предусматривает выбор пиксела и изучение смежных с ним пикселов для проверки близости значений яркости сопоставляемых пикселов. Если значения яркости оказываются близкими, то соответствующие пикселы зачисляются в одну группу, чтобы сформировать область. В этом случае область формируется в результате сращивания отдельных пикселов. В более эффективных вариантах метода в качестве отправной точки используются не отдельные пикселы, а разбиение изображения на ряд небольших областей. Затем каждая область проверяется на однородность (равномерность освещенности), и если результат проверки оказывается отрицательным, то соответствующая область разбивается на более мелкие участки. Процесс продолжается до тех пор, пока все выделенные области не выдержат проверку на однородность. После этого начинается формирование областей при помощи наращивания небольших областей, а не пикселов. Особенности реализации этого метода тесно связаны со структурами данных, используемыми для представления изображения. Проблема выделения области путем наращивания будет рассмотрена в гл. 6 в качестве примера применения таких структур. Основным преимуществом использования небольших областей вместо пикселов является уменьшение чувствительности к шуму.

4.4.1. СЕГМЕНТАЦИЯ НА ОСНОВЕ СРЕДНЕГО УРОВНЯ ЯРКОСТИ

Один из критериев однородности области основывается на оценке максимальной разности значения яркости отдельного пиксела и среднего значения яркости, вычисленного по соответствующей области. Пусть для некоторой области R размером N

$$m = \frac{1}{N} \sum_{P \in R} f(P). \tag{4.2а}$$

В таком случае область называется однородной по порогу T, если выполняется условие

$$\max_{P \in R} |f(P) - m| < T. \tag{4.2б}$$

Это определение однородности можно считать эвристическим, однако, введя некоторые допущения, для него можно дать и теоретическое обоснование. Приводимые ниже рассуждения иллюстрируют многие проблемы, возникающие при выделении области путем наращивания; этот анализ можно рассматривать в качестве модели для исследования других критериев однородности области.

Допустим, обрабатываемые нами изображения класса 1 на самом деле представляют собой изображения класса 2, на которые наложен гауссовский белый шум с нулевым средним значением.

Отсюда следует, что вероятность наличия в пикселе P шумовой составляющей z определяется выражением

$$p_n(z) = \frac{1}{\sqrt{2\pi}\sigma} e^{-z^2/(2\sigma^2)}, \qquad (4.3а)$$

где а — среднее квадратическое отклонение шума. Поскольку мы считаем шум белым, значения вероятности $p_n(z)$ абсолютно не зависят от местоположения пиксела P. В физическом смысле это означает, что на все пикселы шум воздействует одинаково. Вероятность отличия значения яркости пиксела P от его среднего значения яркости более чем на величину x определяется выражением

$$I_n(x) = \frac{2}{\sqrt{2\pi}\sigma} \int_x^\infty e^{-z^2/(2\sigma^2)}\, dz \ . \qquad (4.3б)$$

Наличие множителя 2 в этом выражении обусловлено тем, что мы рассматриваем как отрицательные, так и положительные отклонения от среднего. Правую часть выражения (4.3б) называют *интегралом вероятности ошибки* и обозначают *erf(t)*, где $t = z\sigma$. В большинстве сборников математических таблиц приводятся значения этой функции и некоторые из них мы воспроизводим в табл. 4.1.

Таблица 4.1

Некоторые значения интеграла вероятности ошибки

t	1	1,5	2	2,5	3	3,5	4
erf (t)	0,317	0,134	0,046	0,012	0,003	0,0005	0,0001

Если область является однородной, то оптимальная оценочная функция для значения яркости определяется выражением (4.2а) (см. задачу 4.4). Тогда отклонения значений яркости пикселов от значения оценки m будут определяться исключительно шумом и поэтому вероятность невыполнения неравенства (4.2б) для какого-то пиксела указывается выражением (4.3б), если $z=T$. (В этом случае она задается значениями интеграла вероятности ошибки *erf (T/σ)*.) Например, при $T = 2\sigma$ в неравенстве (4.2б) вероятность невыполнения этого условия для некоторого пиксела составляет 4,6%, а при $T = 3\sigma$ — приблизительно 0,3%. Значение этой вероятности будем обозначать *p(T)*. Поскольку вероятность выполнения условия (4.2б) для отдельного пиксела равна 1—*p(T)*, а область, включает N пикселов, то не выделить однородную область мы можем лишь с вероятностью $1—[1—p(T)]^N$. Для значений *p(T)* много меньших *1/N* эта вероятность составляет приблизительно *Np(T)*. Выбрав пороговое значение T равным утроенному среднему квадратическому отклонению шума, для квадратной области

размерами 16x16 (256 пикселов) вероятность отказа при проверке области на однородность получаем равной 54% (при использовании точной формулы). При выборе порога для T равным четырехкратному значению среднего квадратического отклонения указанная вероятность составляет всего 2,5%, что вполне приемлемо при решении реальных задач. (В последнем случае и точная, и приближенная формулы дают близкие результаты — 0,0253 и 0,0256 соответственно.)

Зачисление однородной области в неоднородные — не единственная из возможных ошибок, поэтому теперь необходимо оценить вероятность отнесения неоднородной области к однородным. В последнем случае различия значений оценки m и яркостей области определяются также различиями значений яркости областей изображения класса 2. Пусть m_1 и m_2 — указанные значения и q_n, — доля пикселов области, истинное значение яркости которых равно m_n $(i=1, 2)$. Если соответствующая область достаточно велика для того, чтобы можно было при определении среднего значения пренебречь влиянием шума, то среднее значение равно $q_1m_1 + q_2m_2$. Если истинное значение яркости пиксела равно m_1, то разность между этим значением и оценкой среднего значения выглядит следующим образом:

$$\delta m = m_1 - (q_1 m_1 + q_2 m_2) \qquad (4.4)$$

Итак, событие, состоящее в том, что наблюдаемое значение яркости пиксела отличается от среднего значения $q_1m_1 + q_2m_2$ более чем на T, может иметь место, если оно отличается от истинного значения яркости m_1 на $T \pm \delta m$, причем вероятность любого из двух последних событий определяется как

$$p_1 = \frac{1}{2}\,[P\,(|T - \delta m|) + P\,(|T + \delta m|)]. \qquad (4.5)$$

Таким образом, p_1 представляет собой вероятность невыполнения неравенства (4.2б) для наблюдаемого значения яркости пиксела, истинное значение яркости которого равно m_1. Вероятность того, что ни для одного пиксела подобного нарушения условия (4.2б) не происходит, определяется следующим выражением:

$$p_u = (1 - p)^{q_1 n}\,(1 - p_2)^{q_2 n}, \qquad (4.6)$$

где вероятность p_2 определяется аналогично вероятности p_1. Итак, p_u представляет собой вероятность отнесения области к однородным, тогда как в действительности она таковой не является. Очевидно, если значение δm мало по сравнению с пороговым значением T, то вероятность Pi близка вероятности $p(T)$. Если то же самое справедливо для вероятности p_2, то вероятность p_u приближенно равна $[1 - p(T)]^N$, т. е. вероятности отнесения к однородной области, которая в действительности является однородной. Другими словами, обнаружение однородности области оказывается некоторым случайным событием. Ситуации, в которых δm оказывается малой, обычно возникают в случаях, когда рассматри-

ваемая область состоит почти исключительно из пикселов одного характера, и, следовательно, ошибка отнесения такой области к однородным не столь серьезна. Если, с другой стороны, в рассматриваемой области представлены примерно в равной мере пикселы обоих типов, т. е. если $q_1=q_2$, так что

$$\delta m = (m_1 - m_2)/2, \tag{4.7}$$

то мы предпочли бы, чтобы вероятность отнесения такой области к однородным была очень малой. Это так, если абсолютное значение разности δm предполагается много большим среднего квадратического отклонения шума, так как в этом случае значение δm будет сравнимо с пороговым значением T. При выполнении этих условий аргумент первого члена суммы в выражении (4.5) оказывается близким к нулю и, следовательно, соответствующая вероятность близка к 1. Второй член этой суммы имеет аргумент, равный среднему квадратическому отклонению, взятому с высокой степенью кратности, и, следовательно, соответствующая вероятность оказывается близкой к нулю. В результате введения этих допущений вероятность p_1 составляет приблизительно 0,5 и можно показать, что вероятность p_2 имеет то же самое значение. В таком случае вероятность $p_u \bullet 0{,}5$. В результате невыполнение условия (4.2б) влечет с вероятностью $1-0{,}5^N$ признание области неоднородной. При $N = 256$ эта вероятность очень близка к 1.

4.4.2. ДРУГИЕ КРИТЕРИИ ОДНОРОДНОСТИ ОБЛАСТИ

Проведенный анализ показывает, что критерий однородности области, определяемый выражением (4.2б), сопряжен с вероятностью чрезмерно частого объявления изучаемой области неоднородной. Это обстоятельство может, по крайней мере отчасти, объяснить, почему сегментация изображения с помощью критерия типа (4.2б) обычно приводит к появлению значительного числа небольших областей, судя по всему не имеющих каких бы то ни было реальных прототипов на изображении. Доскональный анализ иных критериев, так же как и обсуждение влияния размера области (см. разд. 4.5), выходит за пределы задач нашей книги. Несколько иное определение однородности возникает при сопоставлении статистических характеристик, подсчитанных для некоторой области и для отдельных ее частей. Если значения этих статистик оказываются близкими, то соответствующую область можно считать однородной. Такой подход может быть полезным при сегментации, основанной на изучении *текстуры*. Матрицы совместной встречаемости уровней серого тона можно формировать для каждой группы областей отдельно, после чего сравнивать построенные матрицы между собой. Если матрицы подобны, то объединение соответствующих областей представляет собой однородную область. Рассмотрим общий случай. Пусть $F(R)$ — *признак*, оцениваемый на области R. Если R_{12} является объединением двух смежных, но непересекающихся областей R_1 и

R_2, то критерий однородности можно задать, потребовав, чтобы оценка значения признака $F(R_{12})$ была близка оценкам значений признаков $F(R_1)$ и $F(R_2)$. Пусть R_1 — уже выделенная область, а R_2 — небольшая область, для которой проверяется возможность присоединения к R_1 (другие варианты рассматриваются в гл. 6). Необходимо задать значение порога T' таким образом, чтобы при абсолютном значении разности $F(R_1)$ и $F(R_2)$, меньше T', принималось решение об однородности области. Анализ, аналогичный проведенному в подразд. 4.4.1, показывает, что пороговое значение T' должно быть больше дисперсии значений признака $F(R)$, определяемой шумом. Во многих случаях, однако, эта дисперсия много меньше дисперсии собственного шума, в то же время как дисперсия, определяемая неоднородностью, имеет то же значение, что и выше. Таким образом, критерий однородности, основанный на сопоставлении признаков, много надежнее критерия однородности, задаваемого выражением (4.2б). (Само среднее значение, естественно, является таким признаком.)

4.5. БИБЛИОГРАФИЧЕСКИЕ ЗАМЕЧАНИЯ

Эффективные методы сегментации обычно рассматриваются как процедуры распознавания образов. Более сложные схемы обнаружения границ, методы выделения областей путем наращивания и анализ сцен рассматриваются в работах [3.7, гл. 4—6] и [2.3, в гл. 7 и 8]. В статье [4.1] проведены подробный анализ и сравнение несложных схем обнаружения границ, аналогичных определяемым уравнениями (4.1). В работе [4.3] дано теоретическое обоснование для ряда схем обнаружения границ. Рассмотрение сегментации как задачи оценивания и обсуждение влияния размера области на надежность оценок среднего значения и т. п. можно найти в работе [4.5].

Мы надеемся, что анализ, которому посвящен разд. 4.4, продемонстрировал необходимость в моделях изображений. Последние не следует смешивать с моделями объектов, используемыми в анализе сцен, хотя часто они и могут быть с ними связаны. Модели, используемые при анализе сцен, обычно имеют детерминистский характер, а модели изображений являются по преимуществу вероятностными и основаны на распределении математического ожидания шума и разностях значений статистических характеристик отдельных областей изображения[1]. Проблеме моделирования изображений посвящен журнал *CGIP (Computer Graphics and Image Processing)*, 1980, v. 12. Задачи сегментации рассматриваются, в частности, в статьях [4.4, 4.6, 4.7].

[1] Ни в коем случае не следует считать, что модели изображений обязательно должны быть вероятностными, — автор, очевидно, имеет в виду практику, сложившуюся в обработке изображений Постановка и решение задач распознавания и анализа изображений предполагают интенсивное использование алгебраических, логических и теоретико-множественных методов при синтезе моделей изображения. — *Прим., ред.*

Доклад [4.2] представляет собой обзор методов анализа сцен, в котором основное внимание уделяется трехмерным моделям и моделям освещенности. На этом направлении взаимодействие методологий обработки изображений и машинной графики могло бы быть очень плодотворным, однако эта работа находится еще на самых ранних стадиях.

4.6. ЗАДАЧИ

4.1. Предложите процедуру обнаружения границ для сегментации изображения, приведенного на рис 4 7

4.2. Определение местоположения границ на изображении можно рассматривать как задачу поиска неоднородных областей. Попробуйте применить метод анализа, изложенный в подразд. 4.4.1, для исследования схем обнаружения границ, определяемых выражениями (4.1).

4.3. Одна из альтернатив условию (4.2б) заключается в сравнении значений среднеквадратической ошибки с некоторым пороговым (первая берется вместо максимальной ошибки) Считаете ли Вы такой метод разумным?

4.4. Докажите, что значение m, определяемое уравнением (4.2а), минимизирует интегральное квадратическое отклонение $f(P)$ от m.

Замечание. Это доказательство имеется в большинстве учебников по математической статистике и обработке сигналов, однако его поиск займет больше времени, чем потребуется для самостоятельного вывода

4.5. Воспользуйтесь программой редактирования для организации вычисления статистических характеристик изображения в диалоговом режиме Задайте окно на том участке изображения, который Вам кажется однородным, и вызывайте процедуру вычисления статистических характеристик (Гистограмма представляет собой одну из таких характеристик.)

Глава 5

ПРОЕКЦИИ

5.1. ВВЕДЕНИЕ

В обработке изображений термином *проекция* обычно обозначают отображение изображения в сигнал, значения которого представляют собой суммы значений яркости элементов изображения, расположенных вдоль определенных направлений. (В машинной графике, как и в некоторых задачах анализа сцен, этот термин обычно используется для обозначения отображения трехмерного объекта на плоскость, при котором вся информация о глубине объекта теряется.) *Восстановление* тоновых изображений по их проекциям является одним из наиболее известных способов применения ЭВМ в медицине, поскольку оно представляет собой мощное средство диагностики, заменяющее ряд болезненных и опасных методов обследования (см. разд. 5.5.). Этот метод может быть распространен и на восстановление трехмерных объектов по их двухмерным проекциям при помощи объединения последовательности поперечных сечений. Восстанавливаемым объектом, как правило, является какой-либо из органов тела человека (сердце, мозг и т. п.), а проекции представляют собой радиографические или ультразвуковые изображения, однако собственно теория применима к объектам и проникающим излучениям любых типов. Результаты такого восстановления представлены на рис. 5.1. В этой главе будут рассмотрены некоторые наиболее простые методы восстановления, причем их использование в медицине затрагиваться не будет. Методы, имеющие фундаментальное значение для подобных процедур восстановления изображений, будут изложены в разд. 5.2 и 5.3.

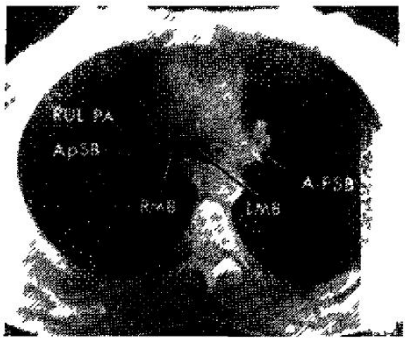


Рис. 5.1. Результат восстановления поперечного сечения тела человека на уровне верхней части грудной клетки — на восстановленном изображении видны бронхи.
Метка RUL-PA обозначает легочную артерию, другие метки обозначают бронхи (Воспроизводится из статьи NaidichD. P., Terry P B, Stitik F. P. and Siegelman S. S Computed Tomography of the Bronchi: 1. Normal Anatomy —*Journal of Computer Assisted Tomography,* 1980, v. 4, p. 746—753)

Проекции использованы также для *анализа формы объектов:* эти прикладные задачи обсуждаются в разд. 5.4. И, наконец, в приложении 5.А приведена распечатка программы восстановления изображений, в которой для вывода тонового изображения используется печать с наложением; программа предназначена для реализации на универсальной ЭВМ.

5.2. ВВЕДЕНИЕ В МЕТОДЫ ВОССТАНОВЛЕНИЯ ИЗОБРАЖЕНИЙ

Если $d(x, y, z)$—плотность некоторого объекта в точке с координатами x, y, z и поглощение рентгеновских лучей пропорционально этой плоскости с коэффициентом пропорциональности k, то функция яркости рентгеновского снимка (негатива), параллельного плоскости (y, z), имеет вид

$$p(y, z) = k \int_L f(x, y, z)\, d(x, y, z), \tag{5.1}$$

где L — прямая, проходящая через объект.

Если направление рентгеновского излучения не совпадает с осью x, а образует с ней угол φ (рис. 5.2), то интегралы должны браться вдоль прямых, определяемых уравнениями вида

$$x \sin \varphi - y \cos \varphi = t, \tag{5.2}$$

где t — евклидово расстояние прямой от начала координат. В таком случае

$$p(\varphi, t, z) = k \int_v d(x, y, z)\, \delta [x \sin \varphi - y \cos \varphi - t]\, dx dy, \tag{5.3}$$

где v — объем, в котором содержится объект, а символ δ [ ] обозначает дельта-функцию [1]. Введение дельта-функции позволяет определить прямую, по которой берется интеграл, в неявном виде вместо ее задания в явном виде в соответствии с уравнением (5.1).

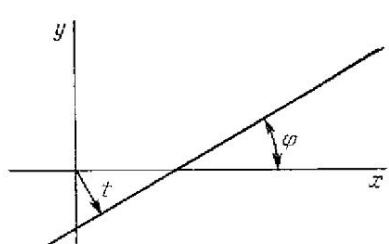


Рис. 5.2. Определение переменных φ и t, входящих в уравнение (5.2)

Фактически проекции строятся лишь для фиксированного значения z, а *поперечное сечение* объекта восстанавливается и отображается. (Повторение такой процедуры при различных значениях z, естественно, приводит к получению информации об объекте в целом.) Поэтому зависимость от координаты z можно исключить

из уравнения (5.3), и обозначив поглощение рентгеновского излучения в точке с координатами (x, y) через $f(x, y)$, получаем

$$p(\varphi, t) = \int_R f(x, y)\, \delta [x \sin \varphi - y \cos \varphi - t]\, dxdy \qquad (5.4)$$

В последнем уравнении через R обозначен участок, содержащий искомое поперечное сечение. Поскольку мы считаем, что вне объекта $f(x, y) = 0$, точная форма участка R не имеет значения.

Восстановление функции $f(x, y)$ по некоторому набору проекций $p(\varphi, t)$, полученных при различных значениях угла φ, основывается на том математическом факте, что двухмерный фурье-образ функции $f(x, y)$ можно получить, располагая набором одномерных фурье-образов проекций $p(\varphi, t)$. Это означает, что $f(x, y)$ можно определить с помощью обратного преобразования Фурье. Это утверждение легко доказывается следующим образом.

Пусть $P(\varphi, w)$ есть фурье-образ проекции $p(\varphi, t)$, определяемый уравнением

$$P(\varphi, w) = \int_{-\infty}^{\infty} e^{-iwt}\, p(\varphi, t)\, dt. \qquad (5.5)$$

[1] Дельта-функцией называется функция, равная нулю всегда, за исключением случая, когда равен нулю ее аргумент, и интеграл которой равен единице в пределах от $-\infty$ до ∞.

Подставляя в уравнение (5.5) уравнение (5.4), находим

$$P(\varphi, w) = \int_{-\infty}^{\infty} \int_{-\infty}^{\infty} f(x, y) \exp[-i w (x \sin\varphi - y \cos\varphi)] dx dy. \quad (5.6)$$

Здесь использовано то обстоятельство, что подынтегральная функция равна нулю всегда, за исключением случая, когда $t = x\sin(\varphi) - y\cos(\varphi)$. Сравнение уравнений (5.6) и (2.2) показывает, что первое представляет собой двухмерное преобразование Фурье функции $f(x, y)$, вычисляемое для частот $w \sin\varphi$ и $-w\cos\varphi$. Можно показать, что обратное преобразование Фурье для двухмерного случая определяется уравнением, аналогичным уравнению (2.6), и в частности

$$f(x, y) = \frac{1}{4\pi^2} \int_{-\infty}^{\infty} \int_{-\infty}^{\infty} F(u, v) \exp[i(ux + vy)] du dv. \quad (5.7)$$

Фурье-образ $P(\varphi, w)$ представляет собой фурье-образ $F(u, v)$, переведенный в полярные координаты, причем $u = w\sin\varphi$ и $v = w\cos\varphi$. При переходе от декартовых к полярным координатам дифференциал $dudv$ заменяется на $|w| dw d\varphi$, где $|w|$ — якобиан матрицы преобразования. Таким образом,

$$f(x, y) = \frac{1}{4\pi^2} \int_{0}^{\pi} d\varphi \int_{-\infty}^{\infty} P(\varphi, w) \exp[i w (x \sin\varphi - y \cos\varphi)] |w| dw \quad (5.8)$$

Математические основы метода восстановления изображений по проекциям сводятся к уравнениям (5.4), (5.5) и (5.8). С теоретической точки зрения можно добиться «идеального» восстановления, вычисляя значения $p(\varphi, t)$ для достаточно большого числа значений φ. Однако при этом возникает много практических затруднений. Прежде чем перейти к их обсуждению, рассмотрим дискретные аналоги указанных уравнений, поскольку при решении реальных задач используются именно они. Уравнение (5.4) изменений не претерпевает, так как процедура проектирования всегда имеет аналоговый характер. Уравнение (5.5) принимает вид

$$P(\varphi, w) = \sum_{k=0}^{N-1} \exp\left(-i \frac{2\pi}{N} wk\right) p(\varphi, k), \quad w = 0, 1, \ldots, N-1, \quad (5.9)$$

где N — число точек, полученных в результате квантования прямой, перпендикулярной плоскости проекции. Уравнение (5.7) принимает вид

$$f(k, l) = \frac{1}{4\pi^2} \sum_{u=0}^{N-1} \sum_{v=0}^{N-1} F(u, v) \exp\left[i \frac{2\pi}{N} (uk + vl)\right]. \quad (5.10)$$

Уравнение (5.9) обеспечивает получение значений фурье-образа $T(u, v)$ в виде $F(w\sin\varphi, -w\cos\varphi)$. Данный способ не позволяет

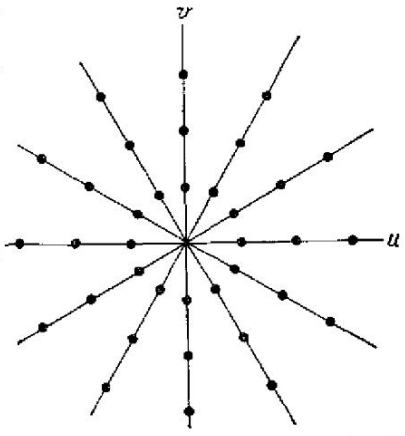


Рис. 5.3. Распределение вычисляемых значений фурье-образа. * - выборочные точки равномерно расположены вдоль лучей, однако в системе координат u-v распределение не является равномерным

получать значения I для точно N^2 равномерно расположенных значений ее аргументов, что иллюстрирует рис. 5.3. Низким частотам соответствует много больше значений, чем высоким. В непрерывном случае [см. (5.8)] эта нехватка высокочастотных значений компенсируется введением коэффициента w и, следовательно, целесообразно использовать не уравнение (5.10), а какой-либо дискретный аналог уравнения (5.8). Большая часть исследований в области алгоритмов восстановления изображений посвящена, в сущности, попыткам найти должную компенсацию неравномерности распределения получаемых значений фурье-образа.

5.3. ОПИСАНИЕ ОДНОГО КЛАССА АЛГОРИТМОВ ВОССТАНОВЛЕНИЯ ИЗОБРАЖЕНИЙ

Проблемы, возникающие при практическом осуществлении восстановления изображений, связаны главным образом с необходимостью быстрого вычисления, чтобы результаты можно было воспроизводить в реальном или почти реальном масштабе времени и обеспечивать низкие уровни шума и искажений. Рассмотрим один распространенный метод восстановления изображений — *метод свертки.*

Пусть внутренний интеграл в уравнении (5.8) определен следующим образом:

$$q(\varphi, t) = \frac{1}{2\pi} \int_{-\infty}^{\infty} P(\varphi, w)|w| e^{iwt}\, dw, \tag{5.11}$$

где

$$t = x \sin\varphi - y \cos\varphi. \tag{5.12}$$

Тогда

$$f(x, y) = \frac{1}{2\pi} \int_{0}^{\pi} q(\varphi, x \sin\varphi - y \cos\varphi)\, d\varphi. \tag{5.13}$$

Для фиксированного значения φ выражение $q(\varphi, x \sin\varphi - y \cos\varphi)$ можно интерпретировать как *обратную проекцию,* т. е. функцию двух переменных, определяемую по некоторой функции

одной переменной, как показано на рис. 5.4. Следовательно, результат восстановления равен разделенному пополам среднему значению обратных проекций, проинтегрированных по φ. [Коэффициент 1/2 появляется вследствие того, что правая часть уравнения (5.13) делится на 2л, тогда как интеграл берется лишь от

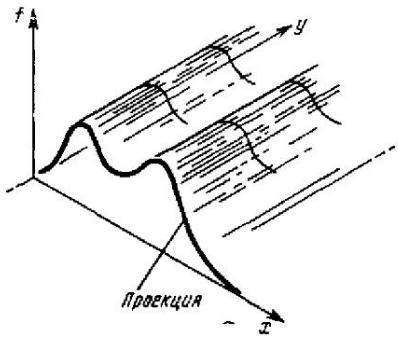


Рис. 5.4. Определение обратной проекции. Кривая в плоскости *f-x* переносится вдоль оси *y* для образования некоторой цилиндрической поверхности, которая рассматривается как некоторая функция двух переменных *(x и y)* и представляет собой обратную проекцию

О до п.] Функция $q(\varphi, t)$ представляет собой обратное фурье-преобразование от $P(\varphi, w)/|w|$, т. е. свертку $p(\varphi, t)$ с функцией $r(t)$, обратное преобразование Фурье которой дает $|w|$:

$$q(\varphi, t) = \int_{-\infty}^{\infty} p(\varphi, \tau)\, r(t-\tau)\, d\tau. \qquad (5.14)$$

Из теории линейных систем известно, что уравнением (5.14) определяется выходной сигнал фильтра, обладающего импульсной характеристикой $r(t)$, при подаче на его вход сигнала $p(\varphi, t)$ (угол φ в данном случае выполняет роль параметра). В результате получаем алгоритм восстановления изображений 5.1 (при его записи через M обозначено число измеряемых значений угла φ, а через $f_r(x, y)$ — восстанавливаемая функция [1]).

Алгоритм 5.1. Стандартный алгоритм восстановления изображений

Обозначения. $p(\varphi, t)$—проекция; $q(\varphi, t)$ — проекции, подвергнутые фильтрации; $f_r(x, y)$ — восстанавливаемая функция; M — число значений угла, использованных при вычислении значений проекций.

1. Установка начального значения $f_r(x, y)=0$.
2. For каждого измеренного угла φ do:
   Begin.
3. Фильтрация $p(\varphi, t)$, в результате которой определяется $q(\varphi, t)$.
4. For каждой пары (x, y) **do:**
   Begin.

[1] Мы вводим здесь другое обозначение, чтобы подчеркнуть неидентичность восстанавливаемой в процессе вычисления функции и истинного поглощения излучения $f(x, y)$.

5. $f_r(x, y) = f_r(x, y) + q(\varphi), x \sin\varphi - y\cos\varphi)$.

End.

End.

6. For каждой пары (x, y), **do:**

Begin.

7. $f_r(x, y) = -f_r(x, y)$.

End. 8.

Конец алгоритма.

Теперь остается только выбрать фильтр, и именно на этом этапе возникают трудности при решении практических задач. Дело в том, что не существует практически реализуемого фильтра с передаточной функцией $|w|$, применение которого не приводило бы к усилению шума. Можно считать, что решающим элементом любого метода восстановления изображений является выбор импульсной характеристики $r(t)$. Приведем некоторые возможные варианты выбора.

а. Импульсная характеристика $r(t)$ представляет собой дельта-функцию, и передаточная функция фильтра равна не $|w|$, а 1. В этом случае нет необходимости использовать операцию свертки, так как $q(\varphi, t) = p(\varphi, t)$. Оказывается возможным просто определить обратные проекции по проекциям $p(\varphi, t)$ и вычислять результат восстановления как среднее последних. Восстановленное изображение, естественно, будет искажено — искажение можно оценить с помощью отношения передаточных функций реального и идеального фильтров. В нашем случае это отношение равно $1/|w|$, т. е. реализуется фильтр нижних частот, и, следовательно, восстановленное изображение оказывается размытым.

б. Обозначим через a расстояние между параллельными лучами проекции, тогда

$$r(0) = 4/(\pi a)^2; \qquad (5.15а)$$

$$r(ka) = -4/(\pi^2 a^2 (4k^2 - 1)), \quad k = \pm 1, \pm 2, \ldots, \qquad (5.15б)$$

а между этими точками $r(t)$ — линейная функция. В результате

$$R(w) = \left| \frac{2}{a} \sin\frac{wa}{2} \right| \left[\frac{\sin(wa/2)}{wa/2} \right]^2. \qquad (5.16)$$

Если $|w|$ много меньше $1/a$, то

$$R(w) \approx |w|. \qquad (5.17)$$

Итак, восстановление $f_r(x, y)$ будет точным, если фурье-образ $F(u, v)$ равен нулю при $|u|$ и $|v|$, больших некоторой малой доли $1/a$. В этом случае также достигается упрощение вычислений. Действительно, наш алгоритм эквивалентен вычислению по следующей формуле:

$$f_r(x, y) = \frac{a}{2M} \sum_{l=0}^{M-1} \sum_{k=-N}^{N} p\left(ka, \frac{l\pi}{M}\right) r\left(x \sin \frac{l\pi}{M} - y \cos \frac{l\pi}{M} - ka\right) \quad (5\ 18)$$

Предполагается, что квантование осуществляется равномерно по углу и расстоянию, а значение N определяется размерами участка, содержащего соответствующее поперечное сечение.

Известно много разновидностей этих методов, однако их обсуждение выходит за рамки нашей книги Вместо их изложения приведем простой пример реализации процесса восстановления.

Пример 5.1. Пусть поперечное сечение представляет собой круг радиусом R я имеет плотность, равную единице В таком случае его проекция при любом значении угла φ будет равна

$$p(\varphi, t) = 2\sqrt{R^2 - t^2} \quad \text{при } t^2 \ll R^2, \quad (5\ 19а)$$

$$p(\varphi, t) = 0 \quad \text{в остальных случаях} \quad (5\ 19б)$$

Отметим, что проекция $p(\varphi, t)$ действительно не зависит от φ Восстановление по двум обратным проекциям, расположенным под углом 90°, без использования фильтрации приводит к результатам, определяемым следующими соотношениями (области, в которых эти соотношения выполняются, указаны на рис. 5 5).

$$|x| \leqslant R \quad \text{и} \quad |y| > R \quad (\text{область } 1) \qquad f_r(x, y) = \sqrt{R^2 - x^2}, \quad (5\ 20а)$$

$$|x| > R \quad \text{и} \quad |y| \leqslant R \quad (\text{область } 2) \qquad f_r(x, y) = \sqrt{R^2 - y^2}, \quad (5\ 20б)$$

$$|x| > R \quad \text{и} \quad |y| > R \quad (\text{область } 3) \quad f_r(x, y) = 0, \quad (5\ 20в)$$

$$|x| \leqslant R \quad \text{и} \quad |y| \leqslant R \quad (\text{область } 4)$$

$$f_r(x, y) = \left(\sqrt{R^2 - x^2} + \sqrt{R^2 - y^2}\right) \quad (5\ 20г)$$

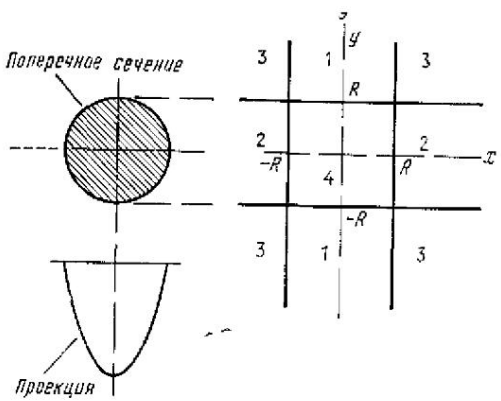


Рис 5 5 Проекция (слева внизу) круглого поперечного сечения (слева вверху) при всех значениях угла Обратная проекция проекции, параллельной оси x, не равна нулю в областях 2 и 4 (справа вверху) Аналогично обратная проекция проекции параллельной оси y не равна нулю в областях 1 и 3

Множитель 2, входящий в уравнение (5 19а), в последних соотношениях опущен, так как берется среднее значение двух обратных проекций Несложные вычисления показывают, что при $x = R \cos \psi$, и $y = R \sin \psi$ восстановленное изображение на окружности $(x^2 + y^2 = R^2)$ определяется с помощью соотношения (5 20г) как

$$f_r(x, y) = R(|\sin \psi| + |\cos \psi|). \quad (5\ 21)$$

При $0<\psi<\pi/2$ последнее соотношение принимает вид

$$f_r(x,\ y) = R(\sin\psi + \cos\psi) = \sqrt{2}R\cos(\psi - \pi/4), \qquad (5.22)$$

тогда как точное значение есть постоянная, равная единице Целесообразно нормировать каждую проекцию, разделив ее значение на площадь A, равную интегралу проекции по / В нашем случае

$$A = 2\int_{-R}^{R}\sqrt{R^2 - t^2}\,dt + \pi R^2, \qquad (5\ 23)$$

поэтому восстановтенное изображение принимает вид

$$f_r(x,\ y) = \frac{\sqrt{2}}{\pi R}\cos(\psi - \pi/4). \qquad (5\ 24)$$

График функции, определяемой уравнением (5.24), представлен на рис. 5.6

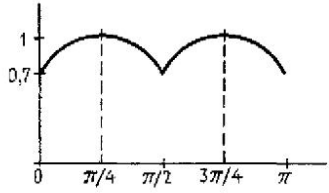


Рис 5 6 Вид функции, восстановленной на границе круга с использованием только двух проекций, по оси ординат откладываются кратные значения *sqrt (2/(πR)*

Приведем еще несколько нормированных значений функции $f_r(x, y)$:

$$f\ (0,\ 0) = 4/(\pi R), \qquad (5\ 25а)$$

$$f(0,\ y) = \frac{2}{\pi R^2}(R + \sqrt{R^2 - y^2}), \quad \text{если } |y| < R, \qquad (5\ 25б)$$

$$f(0,\ y) = 2/(\pi R), \quad \text{если } |y| > R \qquad (5\ 25в)$$

Добавление новых проекций приведет к увеличению значений $f_r(x,\ y)$ в пределах окружности, однако сохранятся ненулевые значения и вне ее, а следовательно, сохранится и размытость восстановленного изображения. Например, введение двух дополнительных проекций изменит значение соотношения (5.25а) на $8/(\pi R)$, хотя значение соотношения (5 25в) изменениям не подвергается.

5.4. ПРИМЕНЕНИЕ ПРОЕКЦИЙ ДЛЯ АНАЛИЗА ФОРМЫ ОБЪЕКТОВ

Проекции можно использовать не только при восстановлении изображений, но также для анализа формы объектов, поскольку построение проекций эквивалентно отображению некоторой двухмерной области в некоторый сигнал Рассмотрим, например, буквенно-цифровые символы — при их написании используются штрихи с небольшим числом направлений ориентации вертикальное, горизонтальное и два диагональных При построении проекции, совпадающей с направлением штриха, последний отражается в проекции как пик, обнаружить который можно легко с помощью целого ряда методов При решении задачи с буквенно-цифровыми символами целесообразно использовать лишь небольшое число проекций, т.е. кроме того, следует вычислять значение интеграла (5.4) непосредственно, вместо того чтобы измерять $p(\varphi,\ t)$

как значение выходного сигнала некоторого реального устройства. Итак, для анализа воспользуемся уравнением (5.4), записав его для угла φ, равного 0, 90, 45 и 135° соответственно:

$$p(0°, t) = \int_R f(x, t)\, dx, \tag{5.26а}$$

$$p(90°, t) = \int_R f(t, y)\, dy, \tag{5.26б}$$

$$p(45°, t) = \int_R f(x, x - \sqrt{2}\, t)\, dx, \tag{5.26в}$$

$$p(135°, t) = \int_R f(x, \sqrt{2}\, t - x)\, dx. \tag{5.26г}$$

Хотя для обеспечения хороших результатов восстановления четырех проекций недостаточно, они позволяют получить существенную информацию относительно штрихов. Одна из проблем, возникающая при использовании проекций в анализе формы объектов, заключается в подборе соответствующего дискретного аналога для уравнений (5.26в) и (5.26г). Алгоритм 5.2 представляет собой простое решение этой задачи, предусматривающее, что все проекции являются векторами с равным числом компонент и размер сетки N — четное число.

Операции, выполняемые на шагах 6 - 13, обеспечивают объединение $2N-1$ диагоналей сетки размерами $N*N$ в N пар, каждая из которых состоит из двух диагоналей (за исключением двух средних, в каждую из которых входит главная диагональ). На рис. 5.7 и 5.8 даны примеры соответствующих бинарных матриц.

Эти примеры показывают, что действительно в проекциях содержится существенная информация о том, как символ выглядит

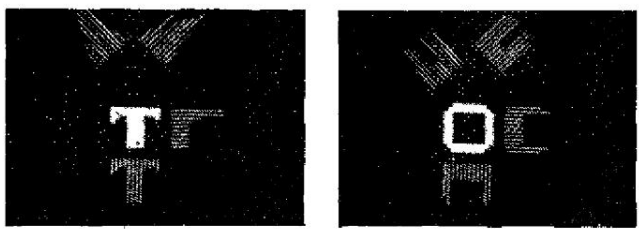


Рис. 5 7. Бинарные матрицы, представляющие напечатанные символы, вокруг которых изображены их соответствующие проекции, найденные с помощью алгоритма 5 2

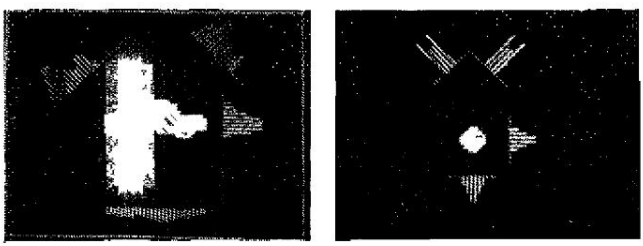


Рис 5 8 Бинарные матрицы, представляющие очертанич автомобиля и диска, вокруг которых изображены их соответствующие проекции, найденные с помощью алгоритма 5 2

в целом. Однако при воспроизведении очертаний автомобиля дело уже обстоит иным образом. Обнаружение пиков проекций можно осуществлять любым способом, принятым в анализе сигналов. Поскольку обычно уровень шума является низким, допустимо применение таких простых методов, как разделение по порогу или даже дифференцирование. Этот метод с заметным успехом используется для распознавания буквенно-цифровых символов даже при построении всего лишь двух проекций (см. разд. 5.5).

Алгоритм 5.2. Определение проекций при анализе формы объектов
Обозначение. $f(i, j)$ — входной массив $N \times N$; $p(k, l)$ — массив *проекций* $4 \times N$.

1. **For** $l=0$ до $N-1$ задается $p(0, l) = \sum_{i=0}^{N-1} f(i, l)$.
2. **For** $l=0$ до $N-1$ задается $p(1, l) = \sum_{i=0}^{N-1} f(l, i)$.
3. Присвоение $p(2, *)$ и $p(3, *)$ начальных нулевых **значений.**
4. **For** $j=0$ до $N-1$ **do:**
   Begin.
5. **For** $i=0$ до $N-1$ **do:**
   Begin.
6. Сопоставление $i+j$ с $N-1$.
7. **If** больше, **then** $f(j, i)$ прибавляется к $p(2, (i+j+1)/2)$.
8. **If** меньше, **then** $f(j, i)$ прибавляется к $p(2, (i+j)/2)$.
9. **If** равно, **then** $f(j, i)/2$ прибавляется к $p(2, (N/2)-1)$ и $f(j, i)/2$ прибавляется к $p(2, N/2)$.
10. Сопоставление $j-i$ с 0.
11. **If** больше, **then** $f(j, i)$ прибавляется к $p(3, (j-i+N)/2)$.
12. **If** меньше, **then** $f(i, j)$ прибавляется к $p(3, (j-i+N-1)/2)$.
13. **If** равно, **then** $f(j, i)/2$ прибавляется к $p(3, (N/2)-1)$ и $f(j, i)/2$ прибавляется к $p(3, N/2)$.
    End.
   End.
14. **Конец алгоритма.**

Для того чтобы визуализировать информацию, содержащуюся в этих проекциях, попытаемся восстановить очертания объекта с помощью обратных проекций, подвергнутых фильтрации. Результаты восстановления представлены на рис. 5.9 и 5.10. Алгоритм восстановления аналогичен алгоритму 5 2 и приведен в виде алгоритма 5.3.

Примеры восстановления, представленные на рис. 5.7.— 5.10, характеризуют ряд достоинств и ограничений, свойственных ме-

тоду. Если контур образован участками окружности, а не прямыми, как, например, в случае очертаний автомобиля, то при восстановлении изображения детали утрачиваются. Однако этот метод именно благодаря низкой чувствительности может оказаться полезен при распознавании символов из разных комплектов шрифта. В самом деле, для выделения штрихов, расположение кото-

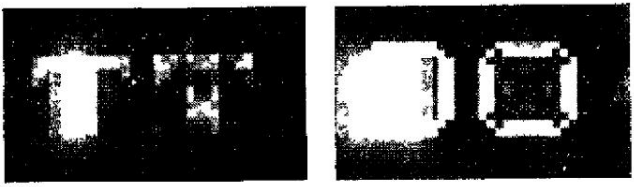


Рис. 5.9. Результаты восстановления изображений объектов, приведенных на рис. 5.7, с помощью обратных проекций, подвергнутых фильтрации

рых совпадает с некоторыми направлениями, можно воспользоваться проекциями. В примере, приведенном ниже, содержится анализ, которым целесообразно пользоваться при определении числа проекций, необходимых для решения конкретных задач.

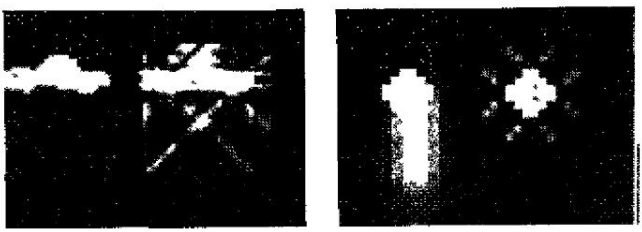


Рис. 5.10. Результаты восстановления изображений объектов, приведенных на рис. 5.8, с помощью обратных проекций, подвергнутых филы - рации

Алгоритм 5.3. Восстановление изображений с помощью обратных проекций

Обозначения. $f(i, j)$ — выходной массив $N \times N$; $p(k, l)$ — массив проекций $4 \times N$.

1. **For** j=0 до $N-1$ **do**:
   Begin.
2. **For** $i=0$ до $N-1$ **do**:
   Begin.
3. $g(j, i)=p(0, j)+p(1, i)$.
4. Сопоставление $i+j$ с $N-1$.
5. **If** больше, **then** $p(2, (i+j+1)+2$ прибавляется к $g(j, i)$.
6. **If** меньше, **then** $p(2, (i+j)/2)$ прибавляется к $g(j, i)$.
7. **If** равно, **then** $1/2[p(2, (N/2)-1)+p(2, N/2)]$ прибавляется к $g(j, i)$.
8. Сравнение $j-i$ с 0.
9. **If** больше, **then** $p(3, (j-i+N)/2$ прибавляется к $g(j, i)$.
10. **If** меньше, **then** $p(3, (j-i+N-1)/2)$ прибавляется к $g(j, i)$.

11. **If** равно, **then** $1/2[p(3, (N/2)-1)+p(3, N/2)]$ прибавляется к $g(j, i)$.

End.

End.

12. **Конец алгоритма.**

Пример 5.2. Допустим, нашими объектами являются исключительно прямоугольники различных размеров и необходимо определить, сколько проекций следует использовать для их обнаружения. Положение прямоугольника относительно направления проекции будем характеризовать с помощью угла φ, образованного большей стороной прямоугольника и перпендикуляром к направлению проекции (рис. 5.11,а). При $\psi=0$ значение проекции равно w на отрезке

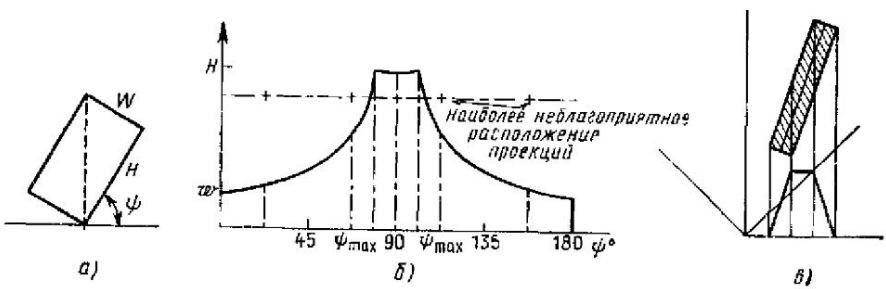


Рис. 5.11. Определение расположения прямоугольника относительно направления проекции с помощью угла ψ (а); изменение максимального значения проекции в зависимости от угла ψ (б); случай, когда значения всех проекций существенно отличаются от максимального размера объекта *(в)*

длиной H, а при $\psi=90°$ значение проекции равно H на отрезке длиной w. При других значениях ψ значения проекции имеют изолированный максимум p_m. Максимальное значение p_m (при изменении ψ) получается в случае, когда направление проекции параллельно диагонали прямоугольника. Значение Ртах = $=sqrt(w^2+H^2)$ достигается при $\psi_{max}=arctg(H/w)$. Здесь w — ширина, H — высота прямоугольника. Функция, характеризующая изменение значений проекций в зависимости от угла ψ, имеет треугольную форму и не равна нулю на отрезке $w \sin\psi + H\cos\psi$ или $2Hw/p_{max}$. При $\psi<\psi_{max}$ $Pm=w/\cos\psi$, а При $\psi_{max}<\psi<90°$ $p_m=H/\sin\psi$. Эта функция симметрична относительно $\psi=90°$ и поэтому ее график имеет вид, представленный на рис. 5.11,б. Очевидно, что при использовании только одной проекции может оказаться невозможным обнаружение прямоугольника при ψ, близких к 90°. При использовании четырех проекций (каждая очередная берется через 45°) максимальное значение искомого пика в наиболее неблагоприятном случае составляет $w/\cos 67{,}5° = 2{,}41\ w$, если $67{,}5°<\psi_{max}$, и $H/\sin 67{,}5°=1{,}08H$ в противном случае. Если $H/w = 5$, то $\psi_{max} = 78{,}7°$, и максимальное значение искомого пика в наиболее неблагоприятной позиции будет составлять примерно половину истинного размера объекта. Последний случай представлен на рис. 5.11, *в*.

Маловероятно, что собственно проекции можно практически использовать для распознавания всех символов алфавита, однако их применение может оказаться полезным в качестве средства предварительной обработки. Ряд символов можно распознать исключительно с помощью их проекций (например, *L, T, I, H* и т. д.), другие требуют дополнительных признаков (например, различения символов *R* и *A)*.

5.5. БИБЛИОГРАФИЧЕСКИЕ ЗАМЕЧАНИЯ

Англоязычная аббревиатура CAT употребляется в случаях, когда речь идет об использовании методов восстановления изображений для построения и воспроизведения поперечного сечения тела человека на основе применения бокового рентгеновского излучения. Первоначально аббревиатура CAT расшифровывалась как *Cross-Axial Tomography* (аксиальная поперечная томография), но сейчас ее читают как *Computer Aided Tomography* (машинная томография). Системы, выполняющие восстановление изображений по проекциям, называют томографическими сканирующими устройствами; в их состав входят мини-ЭВМ, специализированная цифровая аппаратура для выполнения свертки и других преобразований сигнала, графические дисплеи и другое периферийное оборудование ЭВМ, рентгеновская аппаратура и сложные механические приспособления для вращения источника и детекторов рентгеновского излучения вокруг пациента. Появление этих сканирующих систем произвело революцию в медицинской диагностике, поскольку теперь врач располагает изображением; в прошлом для получения необходимой информации приходилось прибегать к хирургическому вмешательству или инъекциям. Некоторое представление о роли томографии в медицине можно составить по статьям, публикуемым в специализированных журналах типа *Journal of Computer Assisted Tomography* (New York: Raven Press). Общая стоимость томографического оборудования велика еще и вследствие того, что для обеспечения параллельности и расположения в одной плоскости пучков рентгеновских лучей необходимо использовать прецезионное механическое оборудование. Именно поэтому проявляется интерес к методам восстановления, допускающим использование непараллельных пучков.

В связи с тем, что восстановление изображений по проекциям имеет колоссальное значение для медицины, большая часть текущей литературы по этой теме связана с медицинскими приложениями метода. Анализ, проведенный в разд. 5.2, основывается на работе [5.9]. Другие примеры задач восстановления изображений, которые решаются описанным методом, можно найти в работе [5.10]. Существует и иная методология восстановления изображений, в основе которой лежит иной принцип, а именно оценивание $f(x, y)$ по значениям проекций можно свести к решению системы линейных уравнений [5.5]. Сборник [5.3] содержит обзоры, касающиеся различных аспектов этой проблемы; краткий обзор, посвященный восстановлению изображений, дан в работе [5.1].

Сборник [5.2], и в частности статья [5.4], посвящен обсуждению ряда актуальных проблем восстановления изображений. Анализ методологии, приведенной в этих работах, обладает общностью, достаточной для того, чтобы заинтересовать и тех читателей, которые не являются узкими специалистами по электронной микроскопии.

Использование проекций позволяет выполнять некоторые разновидности анализа формы объектов, не обременяя себя особыми затратами. Применение проекций при распознавании символов впервые было описано в работе [5.8]. Использовались только вертикальные и горизонтальные проекции, причем после каждого промежутка в символе интегрирование начиналось заново (рис. 5.12). Для определения штрихов использовалось разделение по

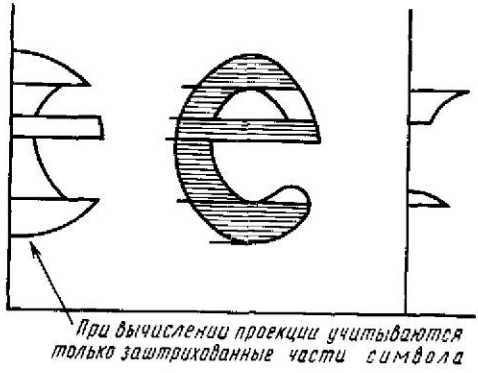


Рис. 5.12. Многосвязные интегральные проекции [5.8], взятые по горизонтали. Если в процессе интегрирования встречается разрыв, то соответствующее значение интеграла рассматривается как первая проекция (слева); результаты последующего интегрирования образуют вторую проекцию (справа)

порогу в сочетании с фиксацией наклона — их наличие в определенных позициях изображения являлось основой для задания двоичных признаков. Далее для получения окончательного классификационного решения применялись статистические методы. Еще один способ использования проектирования описан в докладе [5.6], где рассматривается распознавание китайских иероглифов на основе вычисления фурье-образов четырех проекций. В статье [5.7] описывается использование проекций в сочетании с признаками других типов.

5.6. ЗАДАЧИ

5.1. Реализуйте алгоритм 5.1, взяв в качестве $r(t)$ функцию, определяемую выражениями (5.15) (см. также приложение 5.А).

5.2. Пусть известно, что поглощение излучения $f(x, y)$ представляется в виде $g(x)h(y)$, где g и h — неизвестные функции. Можно ли использовать это свойство для упрощения алгоритма восстановления?

5.3. Дана некоторая функция s, представленная в табличной форме: $N*N$ отсчетов. Продумайте способы использования проекций для уменьшения размеров таблицы. Можно рассмотреть два варианта задачи: в описании функции s допустимо наличие некоторой ошибки и наличие такой ошибки недопустимо.

Указание: следует ввести допущения относительно вида функции и предусмотреть критерии, позволяющие убедиться в том, что функция, представленная заданной таблицей, удовлетворяет этим допущениям.

5.4. Проведите анализ, описанный в примере 5.1, применительно к эллиптической области.

```
$JOB          0660142.EEC5426,T=400,P=500,L=9999
C
C
C       ВОССТАНОВЛЕНИЕ ИЗОБРАЖЕНИЙ ПО ОБРАТНЫМ ПРОЕКЦИЯМ
C
C
C          ЗНАЧЕНИЯ N ПРОЕКЦИЙ ВЫЧИСЛЯЮТСЯ ДЛЯ ИЗОБРАЖЕНИЙ
C          РАЗМЕРА N НА N. ПОСЛЕ ЭТОГО ИЗОБРАЖЕНИЕ ВОССТА-
C          НАВЛИВАЕТСЯ ПО ОБРАТНЫМ ПРОЕКЦИЯМ, ПРИЧЕМ ИС-
C          ПОЛЬЗУЮТСЯ КАК ПРОЕКЦИИ, ПОДВЕРГНУТЫЕ ФИЛЬТРАЦИИ,
C          ТАК И ПРОЕКЦИИ, НЕ ПОДВЕРГАВШИЕСЯ ФИЛЬТРАЦИИ.
C
C          ПРИ ВЫВОДЕ НА ПЕЧАТЬ ВСЕХ ИЗОБРАЖЕНИЙ ПРИМЕНЯЕТСЯ
C          ВЫРАВНИВАНИЕ ГИСТОГРАММЫ.
C
C          ПРОГРАММА НАПИСАНА МАЙКЛОМ КАССОМ   3/17/80
C
C
C
C          ПЕРЕМЕННЫЕ
C
C
C       PIXEL(A,B)    ИСХОДНОЕ ИЗОБРАЖЕНИЕ
C       PIXEL2(A,B)   УЧАСТОК ИЗОБРАЖЕНИЯ, ПОДЛЕЖАЩИЙ ОБРАБОТКЕ
C
C       PROJ(A,B)     ПРОЕКЦИЯ ИЗОБРАЖЕНИЯ ПОД УГЛОМ A*P1/N
C       FPROJ(A,B)    PROJ(A,B), ПОДВЕРГНУТАЯ ФИЛЬТРАЦИИ
C       BPROJ(A,B)    ОБРАТНАЯ ПРОЕКЦИЯ ПРОЕКЦИИ, НЕ ПОДВЕР-
C                     ГАВШИХСЯ ФИЛЬТРАЦИИ
C       FBPROJ(A,B)   ОБРАТНАЯ ПРОЕКЦИЯ ПРОЕКЦИЙ,ПОДВЕРГАВШИХСЯ
C                     ФИЛЬТРАЦИИ
C       AX,AY         КООРДИНАТЫ X,Y НАЧАЛЬНОЙ ТОЧКИ ИНТЕГРИРО-
C                     ВАНИЯ ПРИ ВЫЧИСЛЕНИИ ЗНАЧЕНИЯ ПРОЕКЦИИ
C   d   BX,BY         КООРДИНАТЫ КОНЕЧНОЙ ТОЧКИ ИНТЕГРИРОВАНИЯ
C       P1            СЕРЕДИНА ИЗОБРАЖЕНИЯ
C       P2            СРЕДНЯЯ ТОЧКА ПУТИ ИНТЕГРИРОВАНИЯ
C       P3            ВЕКТОР, ПРОВЕДЕННЫЙ ИЗ P2 В КОНЕЧНУЮ
C                     ТОЧКУ ПУТИ ИНТЕГРИРОВАНИЯ
C
C       CNVRTD        ИСПОЛЬЗУЕТСЯ ДЛЯ ПЕРЕХОДА ОТ СИМВОЛА
C                     К ПРЕДСТАВЛЕНИЮ ЦЕЛЫМ ЧИСЛОМ
C       CNVRT
C
C
C
C
 1      LOGICAL*1 C(6,37),LINE(128),CNVRT(2),FORMS(6)
 2      INTEGER*4 PIXEL(105,128),CNVRTD
 3      INTEGER*4 PIXEL2(64,64)
 4      COMPLEX P1,P2,P3,POLAR
 5      REAL*4 INTBP2
 6      DIMENSION PROJ(64,64),BPROJ(64,64)
 7      DIMENSION FPROJ(64,64),FBPROJ(64,64)
 8      EQUIVALENCE(CNVRTD,CNVRT(1))
 9      COMMON N,SINTB2(64),COSTB2(64)
C
C           ВВОД ИЗОБРАЖЕНИЯ ВО ВНУТРЕННЮЮ ПАМЯТЬ
C
10      DO 2 I=1,105
11      FORMAT(128A1)
12      READ (16,1) LINE
13      DO 2 J=1,128
14      CNVRTD=0
15      CNVRT(2)=LINE(J)
16    2 PIXEL(I,J)=CNVRTD+1
C
C           ВВОД ВО ВНУТРЕННЮЮ ПАМЯТЬ КОДОВ ПЕЧАТИ ИС
```

```
C          НАЛОЖЕНИЕМ
C
17        DO 20 I=1,6
18   20   READ(5,22) FORMS(I),(C(I,J),J=1,37)
19   22   FORMAT(38A1)
C
C             УСТАНОВКА В НАЧАЛЬНОЕ СОСТОЯНИЕ
C
20        N=64
21        N2=N/2
22        DO 39 I=1,N
23        SINTB2(I)=SIN(FLOAT(I-1)*3.141593/FLOAT(N))
24   39   COSTB2(I)=COS(FLOAT(I-1)*3.141593/FLOAT(N))
25        R=.5*N*2**.5
26        P1=CMPLX(FLOAT(N2),FLOAT(N2))
C
C             ВЫБОР УЧАСТКА ИЗОБРАЖЕНИЯ
C
27        DO 14 I=1,N
28        DO 14 J=1,N
29   14   PIXEL2(I,J)=PIXEL(32+I,10+J)
30        CALL PRINT(PIXEL2,C,FORMS,N)
31        WRITE(6,15)
32   15   FORMAT(/15H ИСХОДНОЕ ИЗОБРАЖЕНИЕ)
C
C             ВЫЧИСЛЕНИЕ ЗНАЧЕНИЙ ПРОЕКЦИИ
C
33        DO 67 J=1,N
34        DO 67 K=1,N
C
C           ОПРЕДЕЛЕНИЕ КОНЕЧНЫХ ТОЧЕК ИНТЕГРИРОВАНИЯ
C
35        T=2*R*FLOAT(K-1)/FLOAT(N)
36        P2=P1+POLAR((R-T),J)
37        P3=(0.,-1.)*POLAR((R**2-(R-T)**2)**.5,J)
38        AX=REAL(P2+P3)
39        AY=AIMAG(P2+P3)
40        BX=REAL(P2-P3)
41        BY=AIMAG(P2-P3)
42   68   CONTINUE
C
C             ИНТЕГРИРОВАНИЕ
C
43        SUM=0
44        DO 69 I=1,N
45        X1=AX+(I-1.)*(BX-AX)/(N-1.)
46        Y1=AY+(I-1.)*(BY-AY)/(N-1.)
47   69   SUM=SUM+RINTRP(PIXEL2,X1,Y1,N)
48   67   PROJ(J,K)=SUM
C
C             ФИЛЬТР
C
49        DO 111 I=1,N
50        DO 111 J=1,N
51        FPROJ(I,J)=0.
52        DO 111 K=1,9
53        K1=K-5
54        IF((J+K1).GT.N) K1=N-J
55        IF((J+K1).LT.1) K1=1-J
56   111  FPROJ(I,J)=FPROJ(I,J)-PROJ(I,J+K1)/FLOAT(4*(K-5)**
       $ 2-1)
C
C           ВЫЧИСЛЕНИЕ ЗНАЧЕНИЙ ОБРАТНЫХ ПРОЕКЦИЙ
C
57        DO 106 I=1,N
58        DO 106 J=1,N
```

```
      R1=(((I-N2-1)**2+(J-N2-1)**.5)/(2.**.5)
      IF(J.EQ.(N2+1)) THEN DO
      THETA 1=N2+1
      ELSE DO
      THETA=ATAN((FLOAT(I-N2)-1.)/(FLOAT(J-N2)-1.))
      THETA 1=1.+N*THETA/3.14 1593
      END IF
   91 IF(THETA 1.LT.1) THETA 1=FLOAT(N)+THETA 1
      IF(I.GT.(N2+1)) R1=-R1
      IF((1.EQ.N2+1).AND.(J.GT.N2+1)) R1=-R1
C
C       СУММИРОВАНИЕ ПО ВСЕМ ПРОЕКЦИЯМ
C
      BPROJ(I,J)=0
      FBPROJ(I,J)=0
      DO 106 IPHI=1,N
      R2=R1*COS(FLOAT(IPHI)-THETA 1)*3.141593/FLOAT(N))
      R2=R2+N2+1
      FBPROJ(I,J)=FBPROJ(I,J)+INTRP2(FPROJ,FLOAT(IPHI),R2,N))
  106 BPROJ(I,J)=BPROJ(I,J)+INTRP2(PROJ,FLOAT(IPHI),R2,N)
C
C       ВЫВОД НА ПЕЧАТЬ ИЗОБРАЖЕНИЙ ОБРАТНЫХ ПРОЕКЦИЙ
C
      DO 107 I=1,N
      DO 107 J=1,N
  107 PIXEL2(I,J)=BPROJ(I,J)/FLOAT(N)
      CALL PRINT(PIXEL2,C,FORMS,N)
      WRITE(6,108)
  108 FORMAT(/43H ВОССТАНОВЛЕНИЕ ПО ПРОЕКЦИЯМ, НЕ ПОДВЕР-
     ≠ ГАВШИМСЯ ФИЛЬТРАЦИИ )
      DO 109 I=1,N
      DO 109 J=1,N
  109 PIXEL(I,J)=FBPROJ(I,J)/FLOAT(N)
      CALL PRINT(PIXEL2,C,FORMS,N)
      WRITE(6,110)
  110 FORMAT(/41H ВОССТАНОВЛЕНИЕ ПО ПРОЕКЦИЯМ, ПОДВЕРГАВ-
     ≠ ШИМСЯ ФИЛЬТРАЦИИ )
      WRITE(6,120)
  120 FORMAT(1H1)
      STOP
      END

      ПОДПРОГРАММА PRINT(PIXEL2,C,FORMS,M)
C
C       ПЕРЕМЕННЫЕ
C
C
C       C(A,B)      ДЛЯ УРОВНЯ ЯРКОСТИ В ПЕЧАТЬ С НАЛОЖЕНИЕМ
C                   C(1,B) ПОСРЕДСТВОМ C(6,B)
C       LINE(A)     ИСПОЛЬЗОВАНО В КАЧЕСТВЕ БУФЕРА ДЛЯ
C                   ХРАНЕНИЯ ОДНОЙ СТРОКИ СИМВОЛОВ
C       FORMS(A)    СОДЕРЖИТ '+++++' ДЛЯ ПЕЧАТИ СИГНАЛА
C                   С НАЛОЖЕНИЕМ
C       C1(A,B)     С ДУБЛИРУЕТСЯ В СООТВЕТСТВИИ С ВЫРАВ-
C                   НИВАНИЕМ ГИСТОГРАММЫ, ТАК ЧТО ДЛЯ УРОВ-
C                   НЯ ЯРКОСТИ В C1(1,B) ПОДЛЕЖИТ ПЕЧАТИ С
C                   НАЛОЖЕНИЕМ ПОСРЕДСТВОМ C1(6,B)
C       PIXEL(A,B) СОДЕРЖИТ ЗНАЧЕНИЕ ЯРКОСТИ СООТВЕТСТВУЮ-
C                   ЩЕГО ПИКСЕЛА
C       HIST(A)     СОДЕРЖИТ КОЛИЧЕСТВО ПИКСЕЛОВ СО ЗНАЧЕНИЕМ
C                   ЯРКОСТИ A
C
C
C       R           КАК ОПИСАНО ПАВЛИДИСОМ
C       Z           АЛГОРИТМ ВЫРАВНИВАНИЯ ГИСТОГРАММЫ
```

```
C          HINT
C          HAVE(HAVLIC)
C          LEFT(A)
C          RIGHT(A)
C          NEW(A)
C          MAX          МАКСИМАЛЬНОЕ ЗНАЧЕНИЕ ЯРКОСТИ ПИКСЕЛА
C          MIN          МИНИМАЛЬНОЕ ЗНАЧЕНИЕ ЯРКОСТИ ПИКСЕЛА

      LOGICAL*1 C(6,37),LINE(128),FORMS(6),C1(6,128)
      INTEGER*4 PIXEL2(64,64),HIST(128),LEFT(128),RIGHT(128)
      INTEGER*4 K,I,Z,NEW(128),PIXEL(64,64)
      INTEGER*4 PIX,P1,P2
C
C          ВЫЧИСЛЕНИЕ MAX, MIN ДЛЯ ВЫБОРА МАСШТАБА
C
      MAX=0
      MIN=100000
      DO 23 I=1,N
      DO 23 J=1,N
      PIX PIXEL2(I,J)
      MIN=AMINO(MIN,PIX)
   23 MAX=AMAXO(MAX,PIX)
C
C               ВВЕДЕНИЕ МАСШТАБА
C
      DO 31 I=1,N
      DO 31 J=1,N
   31 PIXEL(I,J)=1+(PIXEL2(I,J)-MIN+1)*126./FLOAT(MAX-
     * MIN+1)
C
C               ПОСТРОЕНИЕ ГИСТОГРАММЫ
C
   62 DO 90 I=1,128
   90 HIST(I)=0
      DO 80 I=1,N
      DO 80 J=1,N
   80 HIST(1+PIXEL(I,J))=HIST(1+PIXEL(I,J))+1
      HAVE=N*N/35
C
C               ВЫРАВНИВАНИЕ ГИСТОГРАММЫ
C
      R=0
      HINT=0
      DO 100 Z=1,128
      LEFT(Z)=R
      HINT=HINT+HIST(Z)
      K=HINT/HAVE
      R=R+K
      HINT=HINT-HAVE*K
      RIGHT(Z)=R
  100 NEW(Z)=(RIGHT(Z)+LEFT(Z))/2
C
C          ДУБЛИРОВАНИЕ МАТРИЦЫ ПЕЧАТИ С НАЛОЖЕНИЕМ
C
      DO 30 I=1,128
      K=I*35/129+1
      DO 30 J=1,6
   30 C1(J,I)=C(J,NEW(I)+1)
C
C             ВЫВОД ИЗОБРАЖЕНИЯ НА ПЕЧАТЬ
C
      WRITE(6,120)
  120 FORMAT(1H1)
      DO 50 I=1,N
      DO 50 K=1,6
      DO 60 J=1,N
```

```
132   60  LINE(J)=C1(K,1+PIXEL(I,J))
133   50  WRITE(6,70) FORMS(K),(LINE(I1=1,N)
134   70  FORMAT(129A1)
135       RETURN
136       END
137       КОМПЛЕКСНАЯ ФУНКЦИЯ POLAR(R,A)
C
C           ОБЕСПЕЧИВАЕТ ПРЕОБРАЗОВАНИЕ ПОЛЯРНЫХ КООРДИНАТ
C           В ДЕКАРТОВЫ   ПРЕДСТАВЛЕНИЕ В ВИДЕ НЕКОТОРОГО
C           КОМПЛЕКСНОГО ЧИСЛА
C
138       COMMON N,SINTB2(64),COSTB2(64)
139       POLAR=CMPLX(R*SINTB2(A),R*COSTB2(A))
140       RETURN
141       END
142       ФУНКЦИЯ RINTRP(MAT,X,Y,N)
143       INTEGER*4 MAT(64,64)
144       INTEGER X1,Y1,A,B,C,D
C           ИНТЕРПОЛЯЦИЯ ПОСРЕДСТВОМ НЕКОТОРОЙ ЦЕЛОЧИСЛЕННОЙ
C           МАТРИЦЫ
C           X,Y   ТОЧКА, В КОТОРОЙ ДОЛЖНО БЫТЬ ВЫЧИСЛЕНО
C           ЗНАЧЕНИЕ ФУНКЦИИ, ПРЕДСТАВЛЕННОЕ МАТРИЦЕЙ
C           N - РАЗМЕР МАТРИЦЫ
C
145       X1=X
146       IF(X1.LE.0) X1=1
147       IF(X1.GE.N) X1=N-1
148       Y1=Y
149       IF(Y1.LE.0) Y1=1
150       IF(Y1.GE.N) Y1=N-1
151       A=MAT(X1,Y1)
152       B=MAT(X1+1,Y1)
153       C=MAT(X1,Y1+1)
154       D MAT(X1+1,Y1+1)
155       E1=A+(B-A)*(X-X1)
156       E2=C+(D-C)*(X-X1)
157       RINTRP=E1+(E2-E1)*(X-Y1)
158       RETURN
159       END
160       ДЕЙСТВИТЕЛЬНАЯ ФУНКЦИЯ INTRP2(MAT,X,Y,N)
161       REAL*4 MAT(64,64)
162       INTEGER X1,Y1,A,B,C,D
C
C           ЭТА ФУНКЦИЯ ТОЧНО ТАКАЯ ЖЕ, КАК ФУНКЦИЯ RINTRP,
C           ЗА ИСКЛЮЧЕНИЕМ ТОГО, ЧТО МАТРИЦА - ВЕЩЕСТВЕННАЯ.
C
163       X1=X
164       IF(X1.LE.0) X1=1
165       IF(X1.GE.N) X1=N-1
166       Y1=Y
167       IF(Y1.LE.0) Y1=1
168       IF(Y1.GE.N) Y1=N-1
169       A=MAT(X1,Y1)
170       B=MAT(X1+1,Y1)
171         MAT(X1,Y1+1)
172       D MAT(X1+1,Y1+1)
173       E1 A+(B A)*(X X1)
174       E2=C+(D C)*(X-X1)
175       INTRP=E1+(E2 E1)*(Y-Y1)
176       RETURN
177       END
        $ENTRY
```

Приложение 5.А. Простая программа восстановления изображений

Ниже приведены распечатка программы на Фортране и пример восстановления изображения, осуществленного в результате ее выполнения Афина была подвергнута дискретизации, затем при помощи суммирования вычислялись значения проекций, после чего проводилось восстановление изображения (с фильтрацией и без фильтрации). Программа написана Майклом Кассой в качестве учебного задания (Принстонский университет)[1]

Я считаю, что такая задача позволяет студентам осмыслить в целом большую часть материала пяти первых глав дискретизацию, выравнивание гистограммы, фильтрацию и т. д. Кроме того, она показывает студенту, что ряд наиболее впечатляющих приложений обработки изображений основывается **на** сравнительно простых принципах

Глава 6 СТРУКТУРЫ ДАННЫХ

6.1. ВВЕДЕНИЕ

Всем, кто сталкивается с обработкой информации, представленной в виде изображений, прекрасно известно, что для ее хранения требуется память с большим объемом. Так, для запоминания, стандартного телевизионного кадра требуется, по меньшей мере, 512x512 байт, если для передачи двух из трех основных цветов используется по 3 бита и для передачи третьего — 2 бита Для запоминания черно белой паспортной фотографии требуется, по меньшей мере, матрица размерами 64x64, на передачу каждого элемента которой затрачивается по 6 бит, что существенно больше размеров записи, содержащей любую иную информацию, которая имеется в паспорте (Для запоминания одной страницы машинописного текста, напечатанного через интервал, требуется около 3000 байт) Проблемы, связанные с хранением, просмотром, поиском, передачей и т. п. , становятся особенно сложными, как только дело доходит до обработки изображений Эти трудности в определенном смысле противоестественны, так как человеку часто легче иметь дело с изображением, чем с текстом Нам значительно проще запомнить лицо нового знакомого, чем страницу машинописного текста Сложность воспроизведения этих способностей человека на ЭВМ можно оценить в полной мере, приняв во внимание, что некоторые люди легче вспоминают лицо, когда речь идет о субъекте противоположного пола, а текст запоминается лучше, если он является частью прозаического произведения, но не списком имен Следовательно, маловероятно, чтобы какие-

[1] Программа реализована на ЭВМ типа IBM 3300 с интерпретирующей программой WATFIV. Полное время выполнения программы составляет 162,5 с. Собственно программа на выходном языке транслятора занимает 17672 байт, а ее массивы — 155915 байт Размеры изображения — 64x64

либо методы сжатия данных, ориентированные на обработку сигналов, позволяли уменьшить объем данных до размеров, соответствующих нашим интуитивным представлениям. Обращение к методам распознавания образов может привести к достижению большей степени сжатия данных (см. пример 1), однако это сопряжено с выполнением достаточно большого объема вычислений и, следовательно, требует преодоления трудностей, связанных с хранением и представлением изображений в памяти ЭВМ.

Задача сжатия данных принимает разные формы в системах связи и обработке информации на ЭВМ. При решении многих прикладных задач некоторое изображение порождается, передается, осматривается и затем ликвидируется. В задачах связи важнейшим требованием является уменьшением *ширины полосы частот,* необходимой для передачи сигнала, при условии, что обработка должна осуществляться за время, сопоставимое с временем, которое приходится затрачивать на порождение и передачу изображения. Иная ситуация возникает при решении прикладных задач, требующих базы изобразительных данных и длительного хранения изображений: приходится сопоставлять новые изображения со старыми или просматривать группы изображений, отыскивая на них определенные признаки. Таким образом, в дополнение к проблемам, связанным с объемами памяти, необходимой для хранения изображений, проблемы могут возникать и при обращении к памяти.

Человеку очень нетрудно сосредоточить внимание на каких-то определенных частях двухмерного изображения, тогда как ЭВМ вынуждена обрабатывать его вслепую. Следовательно, чрезвычайно важны алгоритмы обхода изображения. В дискретном случае обход изображения соответствует обходу некоторой дискретной сетки, которую можно интерпретировать как некоторый граф: его вершинами служат пикселы, а ребра связывают вершины, соответствующие смежным пикселам. Кроме такого простого отображения можно построить и другие графы, соответствующие отображению. Действительно, поскольку все структуры данных, используемые при работе с изображениями, представляют собой графы, целесообразно начать с обзора алгоритмов обхода графов. (Читателям, не знакомым с теорией графов, следует прочесть приложение 6.А). При применении этих алгоритмов к графам, представляющим изображения, действует одно ограничение. Вследствие значительного размера эти графы редко представляют в одной из обычных форм (например, с помощью матрицы смежности, списка вершин и т. д.). В простейшем случае, например, задаются лишь координаты x, y пикселов и их значения яркости или цвета. В результате возникает задача отыскания вершин, смежных с заданной. Если требуется совершить обход только тех участков изображения, которые образуют однородную в некотором смысле область, то приходится решать, принадлежат ли соседние пикселы той же области, что и «текущий» пиксел. Поэтому мы намеренно не будем, по крайней мере в данной главе, точ-

но определять термин «смежный», а будем лишь считать, что существует процедура *ADJACENT (p, n, N)* (СМЕЖНЫЙ), выдающая для вершины (или пиксела) *p* число *n* смежных вершин, которые в процессе обхода еще не посещались, и массив *N*, содержащий описания этих вершин (например, их координаты *x, y)*.

6.2. АЛГОРИТМЫ ОБХОДА ГРАФОВ

Обход связного графа — простая процедура, если она осуществляется следующим образом: выбирается начальная точка, затем осуществляется переход в смежную точку, причем все пройденные позиции отмечаются. Если смежных точек несколько, то выбирается одна из них и в нее осуществляется переход, при этом остальные точки игнорируются — они могут быть пройдены позже при продолжении обхода. Подходящей структурой для хранения этих точек является *стек,* представляющий собой массив, элементы которого накапливаются в процессе обхода, а при обращении алгоритма к массиву элементы из массива удаляются. Основной принцип действия стека состоит в том, что элемент, введенный в него последним, удаляется из него первым. Поскольку впоследствии мы будем многократно обращаться к этим операциям, введем для них специальные назначения: *PUSH(p,S)* (ПРОТАЛКИВАТЬ в СТЕК) — вводить пиксел *p* в стек 5; *p = POP (S)* —удалять точку из стека 5 и обозначать ее символом *p*. В алгоритме 6.1 содержатся две простые программы, реализующие эти функции. В качестве указателя пустого стека используется значение — 1. Если оно оказывается реальным адресом, то должен выдаваться какой-то иной «невозможный» адрес для обозначения пустого стека.

Отметим, что стек S не обязательно должен быть одномерным массивом, особенно если речь идет об обработке изображений. Итак, граф соответствует плоскости изображения, пикселы представляются вершинами графа, причем смежные вершины соответствуют смежным пикселам. Чтобы описать некоторую точку изображения, необходимо запоминать ее координаты *x, y* таким образом, стек 5 оказывается массивом пар (чисел)[1].

Алгоритм 6.1. Процедуры обработки стеков

Обозначения. I — индекс последнего элемента массива *S.*

[1] Подразумевается, что изображение запоминается в виде двухмерного массива. Реализация таких массивов сопряжена со значительными неудобствами при использовании целого ряда машинных языков, поскольку представление данных во внутренней памяти является одномерным. Целесообразно сразу организовывать запоминание изображения в виде одномерного массива, а затем для адресации отдельных пикселов использовать указатели. В этом случае соответствующий стек будет представлять собой массив указателей (см. задачу 6.1).

Процедура *PUSH (p, S)*

1. Увеличение *i* на 1 и задание *S(i)=p.*
2. End.

Процедура, *POP (S)*

1. If i больше 0, **then** *i* уменьшается на 1 и **return** *s(i+1).*
2. **else return** — 1 для того, чтобы обозначить пустой стек.
3. End.

В процессе обхода необходимо использовать два типа разметки. Без потери общности можно предположить, что вначале все вершины (пикселы) помечены единицей. При введении некоторой вершины (пиксела) в стек она помечается двойкой, а после ее прохождения (в процессе обхода) — нулем [1]. Основной алгоритм обхода связного графа (области), в котором используются указанные функции, представлены в виде алгоритма 6.2. Следует также заметить, что можно пренебрегать разметкой пикселов в случае, когда они вводятся в стек. Отказ от разметки не повлияет на «правильность» обхода (см. задачу 6.2), однако может привести к значительному увеличению размеров стека. Можно подобрать примеры, в которых для некоторого графа с N вершинами размеры стека приближаются к N^2. Однако, во многих прикладных задачах граф оказывается таким, что размеры стека не создают проблем (см. подразд. 8.4.4).

Алгоритм 6.2. Основной алгоритм обхода связного графа

Обозначения. Исходными данными для процедуры *ADJACENT (p, n, N)* являются координаты (местоположение) пиксела *p,* а в результате ее реализации определяется число соседних с ним пикселов *n* и их местоположения, которые представляются массивом *N.*

0. Выбор вершины (пиксела) *p* графа, обход которого осущест вляется.
1. Ввод *p в* стек 5.
2. **While 5** непустой **do** шаги 3—9.
   Begin.
3. p = POP(S).
4. **Repeat** шаги *5*—9
   Begin.
5. Присвоение метки пикселу *p* и **call procedure** *ADJACENT (p, n, N).*
6. If *n* равно О, **then** выход из цикла.
7. Задание *p = N(1).*
8. If *n* больше 1, **then do шаг 9.**
   Begin.

[1] Такие метки имеет смысл использовать, в частности, при работе с двухуровневыми изображениями. В этом случае интерес представляют лишь черные пикселы (значение которых равно 1), устраняемые в процессе обхода.

9. Для $i = 2$ до n *PUSH (N(i), S).*
End.

End.

End.

10. **Конец алгоритма.**

Стек представляет собой структуру данных, хорошо подходящую для обхода области, однако существуют задачи, в которых необходимо удалять элементы из массива в той же последовательности, в какой они в него вводились. Соответствующая структура данных называется *очередью.* Функция *ADD (p, Q)* — поставить пиксел *p* в очередь Q — аналогична функции *PUSH,* определенной для стека. Большего внимания заслуживает функция *REMOVE (Q)* (ОЧИСТИТЬ), которая обеспечивает удаление точки из очереди. Эти функции описаны в алгоритме 6.3. Отметим, что для обозначения начала и конца очереди используются два отдельных индекса. Эта структура данных проста, однако, поскольку ее реализация требует определенных затрат памяти, последнюю приходится периодически перераспределять.

Более подробные сведения о методах реализации и обработки стеков и очередей читатель может найти в одной из множества имеющихся книг, посвященных алгоритмам и структурам данных (см. также разд. 6.11)[1]. Если в алгоритме 6.2 заменить стек очередью, то правильность обхода графа не нарушится, хотя порядок обхода вершин изменится. Обход графа можно осуществлять и не прибегая к использованию указанных структур данных — эта возможность связана с применением поиска с возвратом при отсутствии иных способов продолжения обхода. Все упоминавшиеся алгоритмы обхода графа подробно рассматриваются во многих учебниках[2].

Алгоритм 6.3. Процедуры обработки очередей

Обозначения. $i_{кон}$ — индекс последнего элемента массива Q; $i_{нач}$ — индекс первого элемента массива Q. Вначале значения обоих индексов равны первому адресу пространства, занимаемого массивом

Процедура *ADD (p, Q)*

1. Увеличение $i_{кон}$ на 1 и задание $Q(i_{кон})$=P.
2. **End.**

Процедура *REMOVE (Q)*

1. If $i_{нач}$ не больше чем $i_{кон}$ **then** $i_{нач}$ увеличивается на 1 и **return** Q $(i_{нач} - 1)$.
2. **else return** — 1 для того, чтобы обозначить пустую очередь.
3. End.

[1] См., например [6.1, р. 44—49 или 6.8, с 295—381]. [2] См., например, [6.1, р. 176—179 или 3.7, р. 51—57J.

6.3. СТРАНИЧНАЯ ОРГАНИЗАЦИЯ ПАМЯТИ

Использование целочисленной матрицы является простейшим способом представления изображений класса 1. Если размеры матрицы превышают размеры области быстродействующей памяти, отведенной для размещения изображения, то может возникнуть необходимость в разбиении памяти на *страницы*. Вообще говоря, это тривиальная процедура, за тем исключением, что линия, разделяющая страницы, проходит вблизи важной части изображения. Чтобы избежать многократной перекачки страниц, часто прибегают к использованию перекрывающихся страниц На рис 6.1 приведен один из возможных вариантов организации такого перекрытия. Применение перекрытия страниц целесообразно не только с точки зрения минимизации объема перекачки, оно полезно также для выделения таких признаков, которые поддаются обнаружению только в случаях, когда доступна наблюдению достаточно большая область изображения вокруг них.

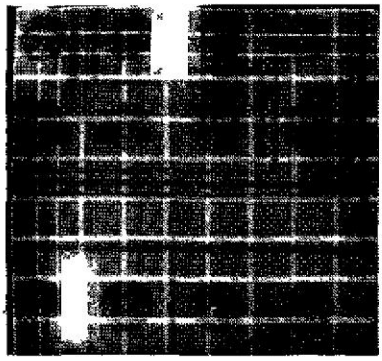
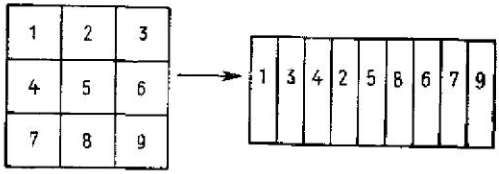


Рис. 6.1. Пример расположения страниц при запоминании изображения перекрытие частей изображения печатной платы

Рис 6 2 Организация расположения страниц памяти, обеспечивающая сохранение геометрической смежности для страницы 5



Эти страницы целесообразно хранить на диске или ленте таким образом, чтобы размещение страниц в памяти геометрически соответствовало расположению страниц на изображении. Можно показать, что в общем случае это условие невыполнимо (см. задачу 6.5). Сохранить геометрическую близость можно лишь для какой-то определенной страницы; на рис. 6.2 приведен соответствующий пример. Важнейшей страницей является пятая, и поэтому все страницы, имеющие с ней общую границу, размещаются в памяти к ней ближе, чем остальные страницы. Такие стратегии могут быть полезны при распределении промежуточной памяти Изображение заносится на ленту, страница вводится в быстродействующую память, причем страницы, расположенные рядом с ней, запоминаются на диске таким образом, как показано на рис. 6.2. Следует иметь в виду, что во многих операционных системах хранятся специфические стратегии распределения памяти, которые, вообще говоря, не обеспечивают сохранения гео-

метрической смежности страниц изображения. В таких случаях необходимо блокировать операционную систему и рассматривать диски в качестве устройств для получения исходных необработанных данных.

6.4. ПИРАМИДЫ ИЛИ ТЕТРАРНЫЕ ДЕРЕВЬЯ

Пирамиды или тетрарные деревья представляют собой структуру данных, которая широко используется и в машинной графике, и в обработке изображений. Ее применение дает наилучшие результаты в случаях, когда изображение представляет собой некоторую квадратную матрицу A, размеры которой определяются некоторой степенью числа 2, скажем, 2^n. Матрицу A можно разбить на четыре квадратные матрицы: A_0, A_1, A_2, A_3, размеры которых в два раза меньше размеров матрицы А. Процесс такого разбиения можно рекурсивно повторить n раз до тех пор, пока не будет достигнут уровень выделения одного пиксела. Эти уровни можно пронумеровать, начиная с нуля, которым обозначается изображение в целом, вплоть до n, соответствующего отдельным пикселам. Каждый выделяемый квадрат можно пометить одним из четырех символов — О, 1, 2 или 3, приписываемых посредством конкатенации к метке породившего его квадрата. В результате отдельные пикселы будут снабжены метками, состоящими из n символов. Эту конструкцию можно представить в виде *дерево*, вершины которого соответствуют отдельным квадратам. Вершины дерева соединяются, если квадрат, соответствующий одной из них, непосредственно содержится в квадрате, соответствующем Другой.

Корень дерева соответствует изображению в целом, листья — отдельным пикселам, а все остальные вершины характеризуются степенью исхода, равной 4. Дерево такого вида обычно называют *деревом четвертой степени* или *тетрарным деревом*. Такое дерево

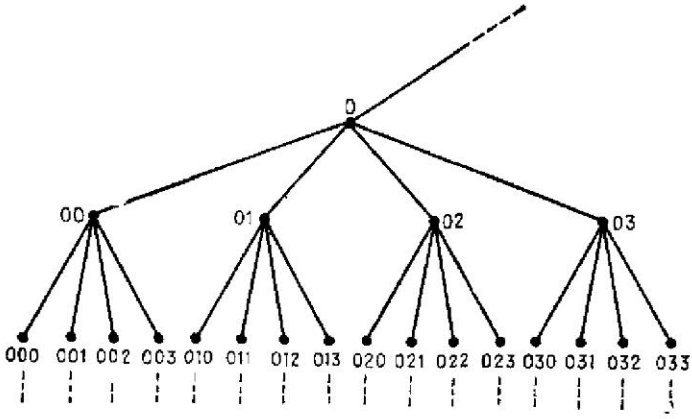


Рис 6 3 Часть тетрарного дерева

представлено на рис. 6.3, а на рис. 6.4 приведена система адресации для изображения размерами 8X8. Поскольку k-й уровень состоит из 4^k квадратов, общее число вершин дерева равно

$$N = \sum_{k=0}^{n} 4^k = \frac{4^{n+1} - 1}{3} \approx \frac{4}{3} 4^n.$$

Следовательно, число вершин приблизительно на 33% превышает число пикселов. Оказывается, что при решении многих прикладных задач для запоминания дерева достаточно использовать только 4^n ячеек. Построение подобного тетрарного дерева будет рассмотрено ниже.

000	001	010	011	100	101	110	111
002	003	012	013	102	103	112	113
020	021	030	031	120	121	130	131
022	023	032	033	122	123	132	133
200	201	210	211	300	301	310	311
202	203	212	213	302	303	312	313
220	221	230	231	320	321	330	331
222	223	232	233	322	323	332	333

Рис. 6.4. Пример, характеризующий систему адресации пикселов изображения размерами 8X8, соответствующую конструкции некоторого тетрарного дерева: первый разряд обозначает вершину первого уровня, второй — вершину второго уровня и т. д.

6.4.1. ПОСТРОЕНИЕ ТЕТРАРНОГО ДЕРЕВА

Допустим, что мы работаем с изображением, выборка значений которого может осуществляться построчно. Такая ситуация имеет место, когда это изображение хранится на ленте или диске, а также когда его дискретизация производится с помощью какого-либо барабанного сканирующего устройства или устройства, выполненного на основе телевизионной камеры (см. разд. 1.3). Для построения соответствующего дерева во внутреннюю память вводятся две строки изображения, после чего начинается просмотр четверок пикселов в последовательности, представленной на рис. 6.5,а Пусть f_0, f_1, f_2, f_3 —соответствующие уровни серого тона для каждой группы, состоящей из четырех пикселов с индексами $2k, 2k+1, 2^n+2k, 2^n + 2k+1$ (k=0, 1, 2, ..., 2^{n-1}-1). Теперь можно определить четыре новых уровня:

$$g_0 = \frac{1}{4} \sum_{i=0}^{3} f_i, \qquad (6.1а)$$

$$g_j = f_j - g_0, \quad j = 1, 2, 3. \qquad (6.1б)$$

Отметим, что при переходе от уровней серого тона f_j к уровням g_j никакой потери информации не происходит, поскольку значения f_j восстанавливаются по значениям g_j при помощи следующих формул:

$$f_j = g_j + g_0, \quad j = 1, 2, 3, \tag{6.2а}$$

$$f_0 = g_0 - \sum_{i=1}^{3} g_i. \tag{6.2б}$$

Единственное затруднение связано с тем, что для представления уровней g_j может потребоваться больше битов, чем для представления уровней f_j. Мы еще вернемся к этому вопросу.

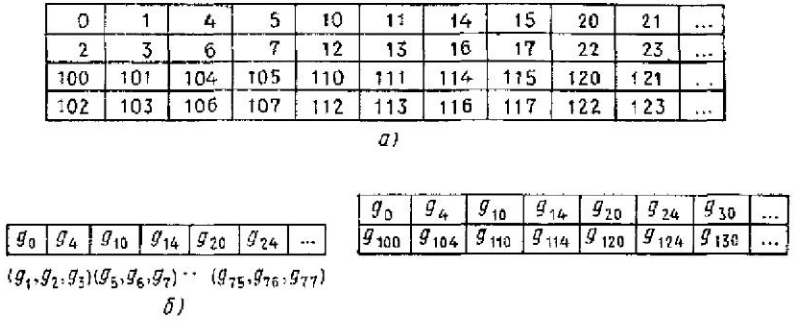


Рис. 6.5. Иллюстрации к построению тетрарного дерева:
а — порядок выборки пикселов из массива при построении тетрарного дерева; метки записаны в восьмеричной системе для изображения размерами 64_{10}X64_{10}; *б* — верхняя строка получена из двух строк исходного изображения, в нижней записан массив, содержащий разности уровней серого тона; *в* — элементы, составляющие две первые строки нового изображения

Для того чтобы сформировать следующий уровень нашего дерева, построим некоторое новое изображение, элементами строки которого служат значения g_0. Одновременно в специальный массив D помещаются значения разностей g_1, g_2, g_3. Таким образом мы приходим к конструкции, представленной на рис. 6.5,*б*. При вводе следующих двух строк индекс первого элемента возрастает на 2^{n+1}, где 2^n — длина строки, и возникает конструкция, приведенная на рис. 6.5,*в*. В результате получим новое изображение размерами 2^{n-1}x2^{n-1} и массив D, состоящий из $3*2^{2n-2}$ значений разностей. Отметим, что, если исходить из номеров пикселов, общий размер памяти тем не менее равен $2^{2n-2}+3*2^{2n-2} = 2^{2n}$. Эта процедура может повторяться до тех пор, пока не будет получено изображение, состоящее из единственного пиксела. При этом почти вся информация будет содержаться в массиве D. Рассмотренный процесс построения можно описать с помощью алгоритма 6.4, в котором используется процедура, представленная в виде алгоритма 6.4а.

Алгоритм 6.4а. Построение уровня тетрарного дерева

Процедура *T. LEVEL (n, I, J, D)*

Обозначения. Входными данными служит изображение / размерами $2^n x 2^n$. В результате выполнения процедуры строится изображение J размерами $2^{n-1} x 2^{n-1}$ и массив D, содержащий значения уровней серого тона.

1. For от 0 до 2^n при увеличении в каждом цикле j на 2 do:
   Begin.
2. Ввод j-й строки изображения / в массив p_1 и $(j+1)$-й строки — в массив p_2.
3. For i = 0 до 2^n при увеличении в каждом цикле i на 2 do:
   Begin.
4. Задание f_0 равного $p_1(i)$, f_1 равного $p_1(i+1)$, f_2 равного $p_2(i)$, f_3 равного $p_2(i+1)$.
5. Вычисление g_0, g_1, g_2 и g_3 по формулам (6.1).
6. Добавление g_0 к изображению J.
7. Добавление g_1, g_2, g_3 к массиву D.
   End.
   End.
8. **Конец процедуры.**

Алгоритм 6.4. Построение тетрарного дерева

0. **Read** изображение I_n размерами $2^n x 2^n$.
1. Задание *уровень* = n.
2. **While** (*уровень* больше 0) **do:**
   Begin.
3. **ВЫЗОВ** ***T. LEVEL (уровень, $I_{уровень}$, $I_{уровень-1}$, $D_{уровень}$).***
4. Уменьшение *уровня* на 1.
   End.
5. **Write** $I_0, D_1, D_2, \ldots, D_n$.
6. **Конец алгоритма.**

6.4.2. ВОССТАНОВЛЕНИЕ ИЗОБРАЖЕНИЯ ПО ТЕТРАРНОМУ ДЕРЕВУ

Если некоторое изображение хранится в памяти в виде некоторой матрицы, то его передача и воспроизведение осуществляются по строкам или линиям развертки. Каждая часть изображения воспроизводится полностью и, следовательно, если T_c — время, необходимое для воспроизведения всего изображения, то за время $T_c/2$ мы получаем лишь его верхнюю половину. Для просмотра изображений такой способ непригоден, особенно если T_c>1—2 с. Предпочтительнее воспроизводить изображение в целом с более низкой разрешающей способностью, но за время $T_c/2$, чтобы можно было быстро определить, имеется ли на нем нечто интересное,

а если нет, то прекратить воспроизведение изображения[1]. Для реализации такого «первоначально грубоинформативного» отображения можно воспользоваться тетрарными деревьями.

Алгоритм 6.5а. Процедура *FILL (i, x_s, y_s, x_e, y_e)*

1. *win (x_s, y_s, x_e, y_e).* {задание на экране окна}
2. *back(i).* {задание цвета фона}
3. *erase.* {приравнивание яркости изображения в плоскости **окна** яркости фона}
4. **Конец процедуры.**

Алгоритм 6.5. Воспроизведение изображения в «первоначально грубоинформативном» виде.

Обозначения. Входные данные представляются массивом *D,* элементами которого служат отклонения от среднего, вычисленного по локальной окрестности, за исключением первого элемента, равного среднему значению, вычисленному по всему изображению.

1. Значение / приравнивается значению первого элемента масси ва *D* и выполняется процедура *FILL (I,* 0, 0, 2^n, 2^n)
2. **For** i **от 0** до *n*—1 **do:**
   Begin.
3. Задается $l=4^n$ и $u = 2^{n-i}$.
4. **For** j = 0 до l—1 **do:**
   Begin.
5. Считывание трех следующих элементов массива *D:* g_1, g_2, g_3.
6. Деление j на 2^n, пусть q — частное, r — остаток. Пусть $y = 2qu$ и $x=2ru$.
7. Считывание значения *go c* изображения в точке *(x, y)* и вычисление f_0, f_1, f_2 и f_3 по формулам (6.2).
8. Вызов процедуры *FILL* (f_0, x, y, $x+y$, $y + u$).
   Вызов процедуры *FILL* (f_1, $x + u$, y, $x + 2u$, $y + u$).
   Вызов процедуры *FILL* (f_2, x, $y + u$, $x+u$, $y + 2u$).
   Вызов процедуры *FILL* (f_3, $x + u$, $y + u$, $x + 2u$, $y + + 2u$).

   End. End. 9.

Конец алгоритма.

Алгоритм 6.5 описывает процесс построения «первоначально грубоинформативного» отображения с помощью тетрарного дерева. В этом алгоритме для воспроизведения изображения используется процедура *FILL* (ЗАПОЛНЕНИЕ), причем предполагается, что изображение может быть получено также с помощью эхо-контроля. Эту процедуру легко реализовать, воспользовавшись элемен-

[1] Не следует считать, что критическим является именно время $T_c/2$, речь может идти и о любой небольшой доле полного времени просмотра изображения.

тарными командами воспроизведения графических объектов, которые были описаны в разд. 1.7 (см. табл. 1.2). Если графический дисплей связан с главной ЭВМ, то этот метод оказывается эффективным лишь при использовании большого окна. При работе с окнами меньших размеров лучше всего буферизовывать выходные данные, формируя в оперативной памяти соответствующую матрицу и осуществляя затем контрольное считывание с помощью команды *wpic* (см. табл. 1.1). При необходимости смены метода критический размер окна определяется исходя из нагрузки на магистраль главной ЭВМ при передаче команд на дисплей, соотношения скоростей выполнения устройством управления дисплеем команды *wpic* и трех простых команд процедуры *FILL,* из объема управляющих операций главной ЭВМ, затрачиваемых на формирование и преобразования буферной матрицы.

6.4.3. СЖАТИЕ ИЗОБРАЖЕНИЙ С ПОМОЩЬЮ ТЕТРАРНЫХ ДЕРЕВЬЕВ

Представление изображений, описанное в двух предыдущих разделах, можно усовершенствовать несколькими способами. В первую очередь необходимо избавиться от увеличения размера памяти (или опасности появления ошибок округления), требующейся для хранения определяемых по формулам (6.1) значений g_j. Необходим метод, обеспечивающий взаимно-однозначное отображение одной четверки чисел в другую.

Утверждение 6.1. Пусть имеется n чисел a_1, a_2, ..., a_n, каждое из которых состоит из Q бит. Предполагается, что n равно степени числа 2. В таком случае для запоминания среднего значения этих чисел и n—1 отклонений от него, обозначаемых соответственно D_2, ..., D_n, достаточно $nQ + log_2 n$ бит.

Доказательство. Пусть L — сумма n чисел, q — частное от её деления на n и r — остаток. Для запоминания каждого из n чисел, включающих q и a_i—q (i = 2, ... , n), требуется максимум Q бит. Для запоминания остатка r требуется максимум $log_2 n$ бит.

Из утверждения 6.1 следует, что при использовании тетрарного дерева каждый уровень требует добавления двух битов, и их накопление, естественно, может превзойти допустимые пределы. С другой стороны, существует определенный баланс между числом битов, затрачиваемых на q, r и a_i—q. Если значения чисел a_i очень близки, то для представления r и a_i—q требуется очень немного битов, однако для хранения q может потребоваться до Q бит. Если различия между значениями чисел a_i велики, то значение q будет малым и для его хранения потребуется меньшее число битов. Это обстоятельство можно использовать для получения представлений, не требующих дополнительных затрат памяти; они рассматриваются в следующем разделе. Здесь мы обсудим, каким образом утверждение 6.1 можно использовать для увеличения эффективности сжатия. Вместо передачи отклонений от среднего значения можно передавать лишь отклонения от q (для представ-

ления каждого из которых требуется максимум Q бит) и последнее q. Кроме того, можно передавать остатки r, для представления каждого из которых требуется всего лишь $log_2 n$ бит (2 бита при использовании тетрарных деревьев). Нетрудно установить, что число остатков равно $(4^n—1)/3$. Таким образом, рост объема памяти будет составлять не 33%, а 33%, умноженные на (2/Q). Для восстановления изображения можно воспользоваться приведенными уравнениями, подвергнутыми очень небольшим изменениям.

Если на изображении имеются большие области с одинаковой яркостью, то можно добиться значительной экономии памяти. Действительно, допустим, что для некоторого участка три разности равны 0. Тогда вместо того, чтобы вводить три нуля в массив D, используемый в алгоритме 6.4а, можно поместить в него некоторый специальный символ. (В качестве последнего можно взять максимально допустимое значение яркости, поскольку оно не может встретиться как некоторое отклонение от среднего значения.) После окончания просмотра всего изображения все повторные вхождения этого символа можно исключить из всех последующих уровней дерева. Так, для кодирования изображения, характеризуемого одним значением яркости, достаточно этого значения и специального символа (см. задачу 6.6).

6.5. БИНАРНЫЕ ДЕРЕВЬЯ ИЗОБРАЖЕНИЙ

Бинарные деревья представляют собой полезную модификацию тетрарных деревьев и обладают рядом достоинств, которые рассматриваются ниже. Хотя этот вид деревьев введен недавно (см. разд. 6.11) и изучен не столь подробно, как тетрарные деревья, бинарные деревья могут в конечном счете оказаться наилучшей структурой данных. На рис. 6.6 представлена блочная организация бинарного дерева. Изменение формы блоков от уровня к уровню не вызывает серьезных затруднений: для ее определения необходимо лишь установить, является индекс уровня четным или нечетным.

Очевидным достоинством бинарных деревьев служит меньшая вариация четкости от уровня к уровню. Если в случае тетрарного дерева разрешающая способность при каждом переходе учетверяется, то в случае бинарного каждый переход приводит лишь к ее удваиванию. Поскольку в обоих случаях общий объем данных остается одним и тем же, бинарное дерево позволяет быстрее построить более точное изображение.

Другое важное преимущество бинарных деревьев — возможность использования соотношения между средним значением и отклонениями, что позволяет передавать не больше битов, чем содержится в исходном изображении. Так, в частности, значения g_0 и g_1 определяются не по формулам (6.1), а с помощью некоторого дискретного приближения, как показано на рис. 6.7.

Последовательность изображений, полученная этим способом, представлена на рис. 6.8. Верхнее крайнее левое изображение ха-

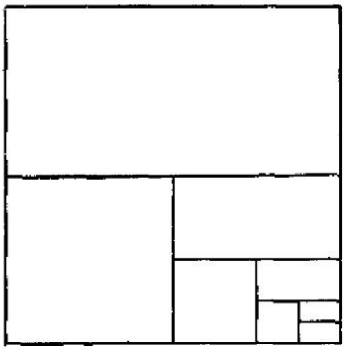


Рис. 6.6 Последовательность представлений изображения с помощью бинарного дерева

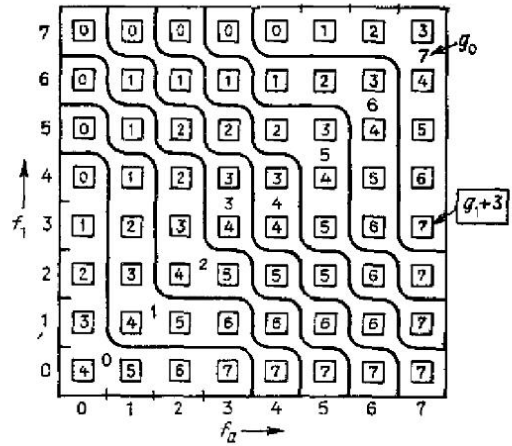


Рис. 6.7. Определение значений g_0 и g_1 по значениям f_0 и f_1.

Числа, помещенные в квадратиках, представляют собой значения g_1, увеличенные на 3 с тем, чтобы они не выходили за пределы диапазона 0—7, эти значения приближенно равны f_0—f_1+3 Квадрат разделен на области в которых числа, не заключенные в скобки, указывают значение go, полученное как приближение среднего значения $(f_0+f_1)/2$

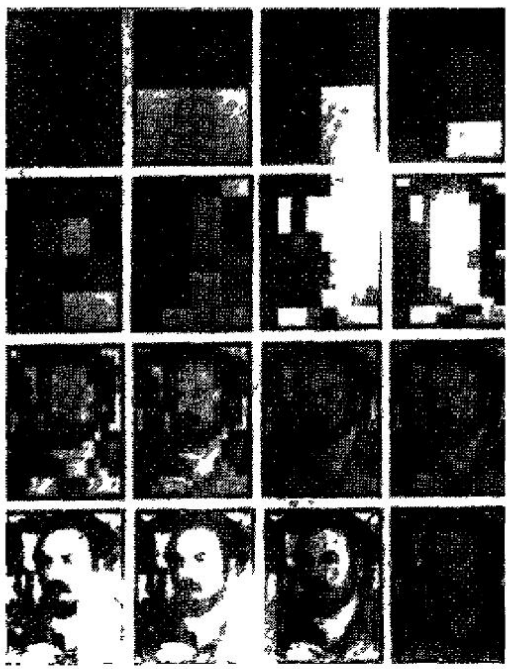


Рис. 6.8. Последовательность изображений, полученная посредством последовательного просмотра элементов бинарного дерева.

(Воспроизводится из статьи Ноултон К Простые, эффективные методы кодирования без потерь для передачи многоуровневых и двухуровневых изображений с постепенным воспроизведением — ТИИЭР, 1980, т 68, № 7, с 149—162)

рактеризуется лишь средним значением яркости по всем пикселам. Первое изображение третьего ряда состоит из 4096 участков (блоков), и на его передачу затрачивается одна восьмая времени, необходимого для передачи последнего (четвертого) изображения, состоящего из 32 768 пикселов. Большая часть информации содержится уже в первом из этих изображений, поэтому можно быстро принять решение о целесообразности продолжения просмотра дерева.

6.6. АЛГОРИТМЫ РАСЩЕПЛЕНИЯ — СЛИЯНИЯ

Сжатие и выборочный просмотр изображений отражают всего лишь два из наиболее типичных способов использования тетрарных (бинарных) деревьев. Еще одну важную область применения эта структура данных нашла в сегментации изображений. В разд. 4.4 были рассмотрены применения процедуры выделения области путем наращивания и процедуры проверки однородности области при сегментации изображений. Алгоритмы расщепления — слияния представляют собой развитие этого подхода. Процесс начинается с изучения вершин (квадратов) некоторого промежуточного уровня тетрарного дерева. Если некоторый квадрат оказывается неоднородным, то он заменяется четырьмя собственными подквадратами (процедура расщепления). С другой стороны, если оказывается, что четыре квадрата образуют однородный квадратный участок изображения, то они и заменяются этим участком (квадратом) (процедура слияния). Процесс может рекурсивно продолжаться до тех пор, пока не возникнет возможность осуществления новых расщеплений или слияний. Эта идея реализуется в алгоритме 6 6, и мы рассмотрим его после того, как определим метод описания вершин тетрарного дерева.

Отметим, что в данном случае дерево используется не для преобразования изображения, а для управления обходом. Так, на уровне q, где квадраты содержат 2^qx2^q пикселов, значение критерия однородности области вычисляется по всем пикселам, а среднее значение по квадрату не используется. Результатом снова является изображение с изменяющейся четкостью, однако в данном случае мы получаем лишь один экземпляр, а не последовательность изображений, как в описанном выше случае. На рис. 6.9 приведен соответствующий пример

Можно построить алгоритмы, отправными точками которых служат отдельные пикселы и которые выполняют только процедуры слияния. Можно построить также алгоритмы, которые применяются к полному изображению и выполняют только процедуры расщепления. Алгоритмы первого типа обладают тем недостатком, что на отдельных пикселах невозможно проверить выполнение многих критериев однородности области. Недостатком алгоритмов второго типа является необходимость выполнения большего объема вычислений, поскольку процедура расщепления более трудоемка, чем процедура слияния (см. задачу 6.8). Использова-

ние в качестве отправной точки промежуточного уровня позволяет вычислять значения сложных предикатов однородности области, обходясь без излишних процедур расщепления.

При реализации этой структуры данных неизбежно возникает проблема, связанная с хранением квадратов с переменными раз-

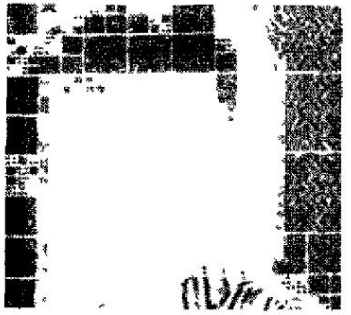


Рис. 6.9. Использование тетрарного дерева в алгоритме расщепления — слияния.

Критерий однородности области основан на вычислении матрицы совместной встречаемости уровней серого тона Границы квадратов наложены на исходное изображение. Первоначально изображение было разделено на 64 квадрата, из которых часть была подвергнута слиянию, а часть расщеплению

мерами. Для каждого квадрата необходимо предусмотреть заголовок, указывающий размеры и положение, выраженные через координаты x, y на плоскости. Если изображение имеет размеры 2^nx2^n, то для представления его размера и положения достаточно затрачивать по n бит. Таким образом, можно ограничиться затратой $3n$ бит на квадрат для описания координат его левого верхнего угла и длины стороны. Можно воспользоваться также системой индексации, описанной в разд. 6.4. В этом случае заголовок будет представлять собой метку, состоящую максимум из n символов. Поскольку в качестве таких символов используется одно из чисел: 0, 1, 2 или 3, то для его представления достаточно иметь два бита. Представление информации о длине метки требует, однако, затраты $\log_2 n$ бит на каждую метку. Для работы с метками с фиксированной длиной можно воспользоваться заполнением свободных мест в метке другим символом, например числом 4, что потребует увеличения затрат памяти до 3 бит на символ. Итак, в зависимости от используемой системы представления на заголовок приходится расходовать $3n$ или $(2n_i + \log_2 n)$ бит, где n_i обозначает длину соответствующей метки. Хотя при использовании заголовков в виде меток с переменной длиной затраты памяти снижаются, соответствующее увеличение сложности реализации, возможно, не компенсируется экономией памяти.

Если число битов, необходимое для представления характеристик квадрата, равно m и число квадратов равно N, то в целом необходимые затраты памяти будут составлять $N(3n + m)$. В случаях, когда для описания области используется много признаков, значение m может оказаться весьма большим. Если, с другой стороны, единственным используемым признаком служит средняя яркость, то значение m равно числу битов, необходимых для описания отдельных пикселов. Значение коэффициента сжатия можно вычислять с помощью следующей формулы: $4^n m/[N(3n + m)]$.

Поскольку типичным значением как n, так и m является число 8, то для представления изображения с переменной четкостью требуется меньший объем памяти, если число квадратов составляет менее одной четвертой числа пикселов. Так как часто дело обстоит именно так, то этот способ не требует дополнительных затрат памяти.

Изображение, получаемое в результате применения алгоритма расщепления — слияния, использующего тетрарное дерево, нуждается в дальнейшей обработке: необходимо слияние областей, не являющихся в данном дереве сибсами. Для осуществления этой процедуры необходимо применять другую структуру данных, которая будет рассмотрена в разд. 6.7.

Алгоритм 6.6 воспроизводит основную процедуру обхода изображения размерами 2^nx2^n в случае, когда просмотр изображения начинается с уровня q, содержащего 4^q квадратов с размерами 2^{n-q}x2^{n-q} каждый. (Таким образом, полное изображение соответствует нулевому уровню, а отдельные пикселы — n-му.) Алгоритм строит не полное дерево, а лишь его необходимые части. Он работает со связным списком L, элементы которого соответствуют вершинам тетрарного дерева, упорядоченным таким образом, что четыре последовательно расположенные вершины имеют на дереве одного и того же предка. Поскольку индекс i_4, используемый при просмотре списка, записывается по основанию 4, координаты x, y левого верхнего угла квадрата определяются непосредственно (см. рис. 6.4). В самом деле, если $d_0, d_1, \ldots, d_k, \ldots$ — разряды индекса i_4, перечисленные, начиная с младшего, то справедливы приведенные ниже уравнения (6.3).

Алгоритм 6.6. Процедуры расщепления — слияния, основанные на использовании тетрарного дерева

Обозначения. Связный список L содержит координаты левого верхнего угла и размеры квадратов, а также указатели на предшествующий и следующий элементы. Индекс q записывается по основанию 4. Входными данными служит набор квадратов, соответствующих некоторому уровню q тетрарного дерева. Выходными данными служит другой набор квадратов, соответствующих другим уровням дерева. Условия **while** на шагах 4 и 11 и условия **if** на шагах 8 и 13 предусматривают вызов какой-либо процедуры проверки однородности области.

1. For i_4 от 0 до 4^q - 1 do: {Установка в начальное состояние}
   Begin.
2. Определение координат x, y по формулам (6.3).
3. Занесение значений x, y и 2^{n-q} в список L
   End.
4. **While** отыщутся области, которые могут быть подвергнуты слиянию, do: {Слияние}
   Begin.
5. Задание индекса j, указывающего первый элемент списка L.

6. **While** будет достигнут конец списка **do:**
Begin.
7. If четыре области, определенные j, j_1 = *следующий* (j). j_2 = *следующий* (j_1) и j_3 = *следующий* (j_2), имеют один и тот же размер, **then do:**
Begin.
8. If их объединение равномерно в смысле используемого критерия, **then** они подвергаются *слиянию* при помощи замены в списке L размером s первой области размером 2s и «отсоединения» трех остальных областей.
9. Замена j на *следующий* (j_3).
End.
10. **Else** замена j на *следующий (j).*
End.
End.
11. **While** отыщутся неоднородные области **do:** {Расщепление}
Begin.
12. **For** всех элементов списка L do:
Begin.
13. **If** данная область однородна, **then** осуществляется продвижение по списку.
14. **Else** замена данной области четырьмя ее дочерними областями.
End.
End.
15. **Конец алгоритма.**

$$x = \sum_k (d_k \pmod 2)\, 2^{n-q+k}, \qquad (6\,3а)$$

$$y = \sum_k \left\lfloor \frac{d_k}{2} \right\rfloor 2^{n-q+k}, \qquad (6\,3б)$$

где символ $[x]$ обозначает наибольшее целое число, меньшее или равное x. Координаты могут принимать значения от 0 до n—— 1. В примере на рис. 6.4 $n = q = 3$, а индекс 213 соответствует

$$x = 1*1 + 1*2 + 0*4 = 3 \text{ и } y = 1*1 + 0*2 + 1*4 = 5.$$

Читатель может убедиться в том, что клетка, помеченная числом 213, действительно расположена на этом рисунке на пересечении четвертого столбца и шестой строки. (К значениям x и y прибавляется по единице, поскольку они равны нулю в первой строке или первом столбце). Таким образом, индекс элемента списка L указывает квадратную область, противоположные (по диагонали) углы которой имеют координаты x, y и $x+2^{n-q}$, $y + 2^{n-q}$. Предпола-

гается, что имеется некоторый критерий однородности области, позволяющий определить, следует ли слить группу областей или расщепить отдельную область. (Подобные критерии были рассмотрены в разд. 4.4.) Алгоритм осматривает область таким образом, что при расщеплении немедленно проверяется однородность ее первой дочерней подобласти.

Описанный алгоритм можно использовать не только для сегментации изображений, но и для решения иных задач. В разд. 17.2 описано его применение для решения задачи разделения видимых и невидимых элементов объекта, однако при этом никаких слияний не производится (см. также задачу 6.9).

6.7. ПОСТРОЧНОЕ КОДИРОВАНИЕ И ГРАФ СМЕЖНОСТИ СТРОК

Если некоторое изображение не удается поместить в оперативную память, то часто его просмотр осуществляется вдоль строк, параллельных заданному направлению (растровая развертка). Этот способ просмотра изображения является также общепринятым в телевизионных системах. Изображение класса 2 можно закодировать, представив его последовательностью длин участков определенного цвета или яркости. Подобную информацию выгоднее всего хранить лишь для первого участка каждой строки, а для остальных участков следует запоминать только отклонения длины и яркости (или код цвета). Такой способ представления изображения часто называют *кодированием с переменной длиной кодовой последовательности* (КПДКП), хотя первоначально этот термин использовался лишь применительно к двухуровневым изображениям, при работе с которыми хранить необходимо только информацию о длине. Этот способ можно использовать и для кодирования изображений класса 1, при этом диапазоны изменения значений уровней яркости аппроксимируются их средним значением. Размер памяти, необходимой для хранения значений длин и яркостей, составляет $I(n + m)$ на одну строку, где I — число интервалов в строке, а n и m обозначают те же величины, что и в предыдущем разделе. При типичном числе уровней серого тона, равном восьми, экономия памяти достигается, если I меньше числа пикселов, составляющих строку, разделенного на два. Таким образом, использование кодирования с переменной длиной кодовой последовательности может привести к существенной экономии памяти, и при таком представлении изображений целесообразно применять алгоритмы анализа изображений и машинной графики непосредственно к закодированным данным, не прибегая к восстановлению исходного изображения[1].

Описание многих алгоритмов обработки изображений, предусматривающих использование КПДКП, упрощается при примене-

[1] Существуют даже дисплеи, в которых предусмотрено использование КПДКП в памяти обновления изображений.

нии *графа смежности строк* (ГСС). Вершины такого графа соответствуют интервалам, а ребра соединяют вершины, если соответствующие этим вершинам интервалы расположены на смежных строках, их проекции по направлению просмотра частично перекрываются и цвет составляющих их пикселов одинаков. На рис. 6.10 приведен пример ГСС. Проекции интервалов $[x_{k-1}, x_k)$ и $[x_{j-1},$

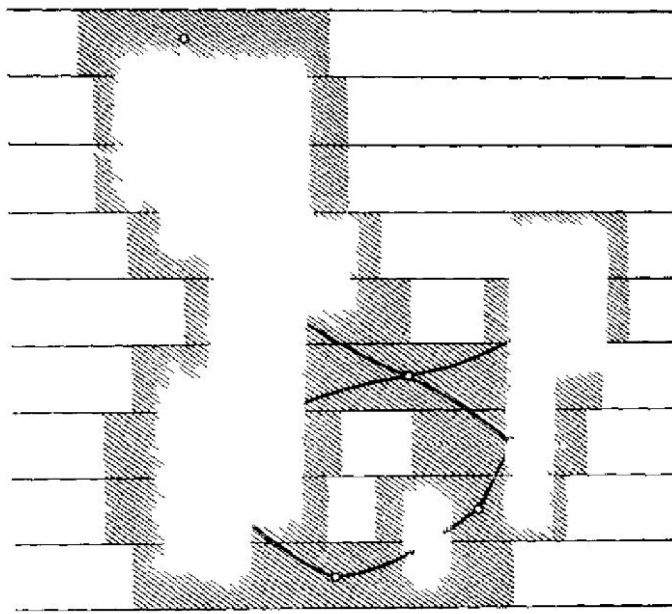


Рис 6.10 Пример смежности строк

$x_j)$ частично перекрываются при одновременном выполнении следующих неравенств:

$$x_{k-1} < x_j \text{ и } x_{j-1} < x_k, \tag{6.4}$$

где x_i — координаты концевых точек интервалов[1].

Запись условий перекрытия в виде строгих неравенств означает, что интервалы, соприкасающиеся лишь одним углом, не считаются перекрывающимися. Это ограничение можно снять, просто взяв нестрогие неравенства. Поскольку условия частичного перекрытия легко проверяются, нет необходимости хранить граф смежности строк в явном виде.

Граф смежности строк можно превратить в ориентированный граф, считая, что ребро ведет из вершины A в вершину B, если интервал, соответствующий вершине A, расположен над ингерва-

[1] Напомним, что запись $[a, b)$ обозначает полуоткрытый интервал, точки, составляющие его, удовлетворяют условию $a<=x<b$.

лом, соответствующим вершине B^1 В таком случае можно говорить о *верхнем* порядке некоторого интервала-вершины (определяется числом интервалов верхней строки, соприкасающихся с данным интервалом) и соответственно его *нижнем* порядке (определяется числом интервалов нижней строки, соприкасающихся с данным интервалом). Для указания порядков вершины ГСС будем пользоваться обозначением *(a, b),* где *a* представляет значение *верхнего,* а *b* — *нижнего* порядков. Нетрудно доказать следующие утверждения, характеризующие некоторые свойства ГСС.

Утверждение 6.2. Если вершина ГСС имеет порядки (0, *d),* то она соответствует локальному максимуму по вертикали. Если вершина имеет порядки *(d,* 0), то она соответствует локальному минимуму по вертикали.

При решении некоторых прикладных задач полезным оказывается следующий результат, относящийся к ГСС, строящимся для нескольких различных цветов Пусть на изображении присутствуют только два цвета, *X* и *Y,* и пусть ГСС для цвета *X* строится в соответствии с неравенствами (6.4), а для цвета *Y* используются уже нестрогие неравенства (допускаются равенства), так что вершины, соответствующие интервалам, соприкасающимся лишь одним углом, оказываются связанными. (Смысл введения этого допущения раскрывается в разд. 7.2.) Соответствующая конструкция представлена на рис. 611.

Утверждение 6.3. Если вершина ГСС, соответствующего цвету *X,* имеет порядки *(m, n), n>1,* то в ГСС, соответствующему цвету У, на строке, смежной снизу со строкой рассматриваемой вершины ГСС для цвета *X,* найдется вершина, имеющая порядки (О, *d).* Аналогичным образом, при *m>1,* в ГСС для цвета У на строке, смежной сверху со строкой рассматриваемой вершины ГСС для цвета *X,* найдется вершина, имеющая порядки *(d,* 0). В обоих случаях справедливы обратные утверждения.

Доказательство. Приведем доказательство утверждения, относящегося к нижнему порядку, большему единицы. Пусть $x_{1,i}$ и $x_{1,i+1}$ — концевые точки интервала цвета *X,* соответствующего вершине, имеющей нижний порядок, больший единицы (см. рис. 6.11,*а*). Это означает, что на смежной снизу строке имеются концевые точки интервалов $x_{2,j-2}$, $x_{2,j-1}$, $x_{2,j}$ и $x_{2,j+1}$, удовлетворяющие следующим неравенствам:

$$x_{1,i} < x_{2,j-1} \text{ и } x_{2,j} < x_{1,j+1} \qquad (6.5)$$

Отметим, что $x_{2,j-1}$ является правой, а $x_{2,j}$ — левой концевой точкой интервалов цвета *X* (см. рис. 6.11,*а*). Следовательно, интервал $x_{2,j-1}, x_{2,j}$ должен иметь цвет *У.* Он будет связан с интервалами того же цвета, находящимися на смежной сверху строке, в том и только том случае, если выполняется одно из следующих неравенств:

$$x_{2,j-1} \leqslant x_{1,i} \text{ или } x_{1,i+1} \leqslant x_{2,j}. \qquad (6.6)$$

[1] В случае ориентированного графа упорядоченная пара вершин (ориентированное ребро) называется дугой, — *Прим. перев*

Поскольку, однако, неравенства (6.6) являются обратными относительно неравенств (6.5), вершина, соответствующая интервалу x_{2j-1}, x_{2j} (цвета У), должна иметь нулевой верхний порядок. (Именно здесь используется допущение о различии определений связности для каждого цвета: обращение нестрогого неравенства

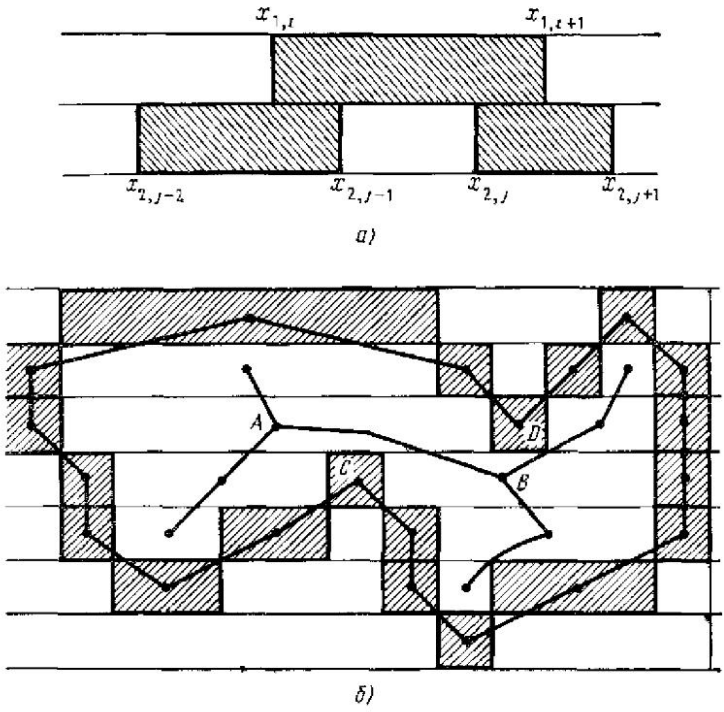


Рис. 6.11. Иллюстрация к утверждению 63.
а — обозначения, используемые при доказательстве утверждения, *б* — общая конструкция

дает строгое неравенство.) Допустим, с другой стороны, что интервал x_{2j-1}, x_{2j} имеет нулевой верхний порядок Из этого следует, что неравенства (6.6) не могут выполняться Таким образом, неравенства (6 5) — истинны, и, следовательно, существует интервал цвета X, нижний порядок которого больше единицы Аналогичное доказательство можно провести для случая верхнего порядка, большего единицы.

Из утверждения 6.3 следует, что интервалы цвета X, у которых *верхний* или *нижний* порядок превосходит единицу, соответствуют тем частям изображения, где область цвета У имеет либо максимум по вертикали, над которым располагается интервал цвета X (например, вершины А и С на рис. 6.11,*б*), либо минимум по вертикали, под которым располагается интервал цвета X (например, вершины B и D на рис. 6.11, *б*).

6.8. КОДИРОВАНИЕ ОБЛАСТЕЙ И ГРАФ СМЕЖНОСТИ ОБЛАСТЕЙ

Данный способ особенно хорошо подходит для представления изображения класса 2 и приближения изображений класса 1. Связную область определенного цвета можно полностью описать с помощью ее контура, причем такое описание обычно требует много меньших затрат памяти, чем описание через пикселы. Обработка представленных таким образом изображений упрощается при наличии графа смежности областей (ГСО). Вершины этого графа соответствуют отдельным областям и ребро соединяет пару вершин, если соответствующие области являются смежными. Следует подчеркнуть, что понятие смежности для дискретных изображений необходимо тщательно определить, чем мы займемся детально в гл. 7. Здесь мы ограничимся замечанием, что какое бы определение смежности областей ни предполагалось использовать, оно не должно допускать соединений, превращающих соответствующий ГСО в неплоский граф. Так, в частности, если четыре области имеют одну точку соприкосновения — угол, то недопустимо введение обоих диагональных соединений, как показано на рис. 6.12. Данное изображение включает пять областей, причем каждая из пар областей *(A, B), (B, D), (D, C)* и (C, *A)* имеет общую границу, и следовательно, соответствующие вершины ГСО должны быть соединены. Кроме того, каждая из вершин *A, B, C и D* соединена с вершиной *E* Если ввести соединения в парах вершин *(A, D)* и *(B, C),* то ГСО становится неплоским. Небольшое изменение положения точки, в которой соприкасаются четыре области, приведет к тому, что сохранится смежность только одной из двух последних пар областей и соответствующий ГСО будет плоским.

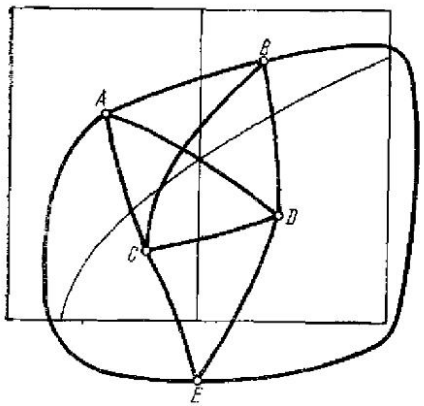


Рис 6 12 Ограничения, налагаемые на определение смежности областей изображения

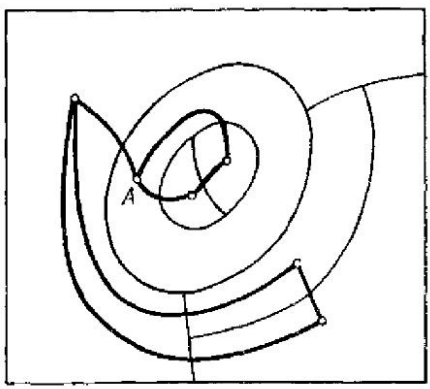


Рис 6 13 Граф смежности областей

На рис. 6.13 приведен пример графа смежности областей, наложенного на изображение. Исследуя свойства ГСО, можно получить существенную информацию об изображении.

Приведем некоторые свойства ГСО и соответствующие им признаки изображений.

а. Порядок вершины определяется числом областей, смежных рассматриваемой области. Это число обычно пропорционально раз меру рассматриваемой области, и, следовательно, вершины, имею щие высокий порядок, соответствуют большим областям.

б. Если внутри некоторой области расположены какие-то дру гие области (например, A на рис. 6.13), то соответствующая ей вершина ГСО является *разделяющей*. Если ни одна из областей изображения не содержит отверстий (все области — односвязные), то ни одна из них не содержится в какой-то другой области и в ГСО нет разделяющих вершин. Поскольку существуют эффектив ные алгоритмы отыскания разделяющих вершин графа (см. разд. 6.11), то всю топологическую информацию об изображении можно извлечь из ГСО.

Вершины ГСО могут быть помечены информацией, характеризующей области, и поэтому, помимо топологических, могут быть изучены и другие свойства. Граф смежности областей можно использовать также для реализации изощренных способов страничного представления изображения. Отдельные страницы составляют части изображения, соответствующие небольшим кластерам вершин (или даже отдельным вершинам), что обеспечивает более естественное, чем при использовании произвольной сетки, разбиение.

И наконец, ГСО является основной структурой данных, используемой при сегментации изображения с помощью выделения областей путем наращивания, в том числе на основе алгоритмов расщепления — слияния. Один из недостатков алгоритма 6.6 состоит в том, что в конечном счете области, обладающие аналогичными признаками, могут не подвергнуться слиянию, если они не имеют в тетрарном дереве общего предка. Использование ГСО на этом последнем этапе позволяет объединять подобные области.

6.9. ИКОНОГРАФИЧЕСКИЕ ПРЕДСТАВЛЕНИЯ [1]

Иконографические представления довольно сложны, однако они являются наиболее эффективным средством хранения и преобразования изображений. При этом способе представления предполагается, что изображение разделяется на подызображения, которыми могут служить произвольные страницы (см, разд. 6.3), квадраты тетрарного дерева, области, полученные в результате сегментации изображения и т. д. Однако вместо того чтобы хранить

[1] Этот термин нельзя отнести к числу стандартных или установившихся понятий обработки изображений; он, очевидно, введен по аналогии с его употреблением в изобразительном искусстве и в искусствознании. Некоторые специалисты по обработке изображений используют термин «иконика», обозначая им как цифровую обработку изображений, так и теорию изображений в широком смысле. — *Прим. перев.*

все изображения, некоторые из них заменяют меткой-описанием. Пример 1.1 иллюстрирует одну из разновидностей иконографического представления. Как правило, получению такого представления должно предшествовать применение какой-нибудь процедуры распознавания образов.

6.10. СТРУКТУРЫ ДАННЫХ ДЛЯ ВОСПРОИЗВЕДЕНИЯ ВИЗУАЛЬНЫХ ОТОБРАЖЕНИЙ

Структуры данных, рассмотренные выше, удобно использовать в случаях, когда работа ведется с собственно изображениями. В машинной графике исходными данными служат описания изображений, причем задаваться они могут не в виде изображений. В общем случае эти описания представляют собой кусочно-полиномиальные функции одной переменной (для кривых) или двух переменных (для поверхностей). При работе с векторными графическими устройствами, элементарные команды которых обеспечивают лишь отображение прямых, эти функции являются кусочно-линейными, что позволяет строить многоугольники и многогранники. В любом случае описание объекта представляет собой некоторый граф, ребрами которого служат отрезки прямых. Говоря об этой структуре данных, можно пользоваться термином *граф объекта.* Если в описание входят отрезки кривых или участки поверхностей высших порядков, ограниченные замкнутыми кривыми, то и в этом случае можно строить граф объекта, но с *помеченными ребрами* — метками служат коэффициенты уравнения кривой или поверхности. Другим возможным вариантом является использование некоторого (неплоского) ГСО, вершины которого соответствуют участкам поверхностей, ограниченным замкнутыми кривыми, и, если эти участки оказываются смежными, соответствующие им вершины соединяются ребрами. Такой «граф смежности участков поверхности», ограниченных замкнутыми кривыми, целесообразно применять при работе с растровыми графическими устройствами.

В принципе, располагая подобными графами, нетрудно строить визуальные отображения. Практической задачей является создание структур данных, которые оказались бы полезными при решении различных задач, возникающих в связи с построением визуального отображения. Так, для увеличения скорости решения задачи разделения видимых и невидимых элементов объекта могут потребоваться проведение предварительной сортировки и введение дополнительных связей. В гл. 17 мы еще вернемся к этим проблемам.

6.11. БИБЛИОГРАФИЧЕСКИЕ ЗАМЕЧАНИЯ

Задачи, связанные с обходом графов и соответствующими структурами данных, рассматриваются в большинстве книг, посвященных алгоритмам или структурам данных (например, [6.1,

6.8]). Обсуждение этих задач в контексте распознавания образов можно найти и в монографии [3.7]. Следует также обратить внимание на работы [6.2, 6.9], в которых обсуждается проблема сохранения геометрической смежности при пословной организации памяти.

Тетрарное дерево — одна из наиболее часто рассматриваемых в современной литературе структур данных. Впервые ее применение приобрело широкую известность в связи с алгоритмом удаления невидимых линий, предложенным Уорноком [6.15]. В начале 70-х годов тетрарное дерево нашло широкое применение в распознавании образов и обработке изображений ([6.4—6.6, 6.13, 6.14]). Использование тетрарных деревьев для получения визуальных отображений методом первоначального грубоинформативного представления впервые было описано в работе [6.12]. Бинарное дерево было не так давно введено Ноултоном [6.7] (рис. 6.8 заимствован из статьи 6.7). Тетрарные деревья являются сейчас популярным объектом исследований, и их свойства изучены достаточно досконально; в докладе [6.11] дан обзор полученных результатов.

Широкое применение нашел также граф смежности строк, хотя часто оно имеет неявный характер. Алгоритмы, предназначенные для выполнения различных операций над изображением, описываются на языке пикселов и интервалов без каких-либо упоминаний о ГСС. Статья [6.3] — это, вероятно, первая работа, в которой списано использование (неявное) ГСС. Другие примеры применения ГСС в распознавании образов и обработке изображений можно найти в монографии [3.7]. В статье [6.10] обсуждается явное использование ГСС при решении задач машинной графики (дальнейшие сведения приведены в гл. 8).

Граф смежности областей является, вероятно, более естественной (и появившейся раньше) для обработки изображений структурой данных; в большей части литературы, посвященной анализу сцен, фигурируют ее различные варианты.

6.12. ЗАДАЧИ

6.1. Один из недостатков использования двухмерных массивов заключается в том, что в некоторых машинных языках их размерность определяется в процессе компиляции. С другой стороны, при использовании одномерного мас сива заранее необходимо определить лишь максимальные размеры массива. Напишите, исходя из этих условий, программу запоминания пикселов некото рого изображения и включите в нее процедуры вычисления значений коорди нат x, y по положению, занимаемому соответствующим пикселом в массиве. Предложите реализацию алгоритма 6.2, основанную на использовании такого массива. Считайте, что смежными для пиксела p являются 8 пикселов, дости жимых из p по одному из направлений, указанных на рис. 1.2.

6.2. Покажите, что правильный обход связной составляющей произвольного графа возможен и в случае, когда одна из вершин графа вводится в стек более одного раза.

6.3. Решите задачу 6.1, предусмотрев следующее изменение алгоритма 6.2: вместо того чтобы вводить в стек пикселы, смежные пикселу p (осматривае мому в данный момент), вводите в стек сам пиксел p Насколько упростит

алгоритм это изменение? Сохраняется ли при этом необходимость дополнительной маркировки пикселов, помещаемых в стек? Требует ли такое упрощение каких-либо дополнительных затрат?

6.4. Напишите программу обхода графа, обеспечивающую определение пар вершин, соединенных маршрутом, который содержит только вершины второй степени. (Этот граф можно рассматривать как сеть связи, в которой вершины первой степени выполняют роль *оконечных станций,* вершины степени больше двух — *аппаратуры обмена данными* и вершины второй степени — *усилителей.* Ваша программа обеспечит выбор всех пар устройств обмена данными и оконечных станций, имеющих непосредственную связь (в том числе через усилители.) Аналогичная задача, относящаяся к обработке изображений, рассматривается в гл. 9.

6.5. Данная задача связана с разбиением памяти на страницы. Рассмотрите множество одинаковых квадратов, расположенных на плоскости таким образом, что они образуют матрицу квадратов размерами NxN. При таком размещении существует $2N(N-1)$ пар квадратов, имеющих одну общую сторону. (На рис. 6.14 приведена соответствующая матрица для $N=4$.)

1	2	3	4
5	6	7	8
9	10	11	12
13	14	15	16

Рис. 6.14. Размещение квадратов в соответствии с условиями задачи 6.5

Необходимо найти такое линейное размещение этих квадратов, которое в максимально возможной степени обеспечит сохранение геометрической близости, соответствующей двухмерному случаю. Проанализируйте различные стратегии, обеспечивающие достижение этой цели, а также различные варианты задания меры близости Так, при расположении квадратов в соответствии с их номерами (приведенными на рис. 6.14) непосредственная смежность сохраняется для $N(N-1)$ пар Зигзагообразное расположение обеспечивает сохранение непосредственной смежности для $(N-1)(N+1)$ пар. Существует ли стратегия, •обеспечивающая сохранение непосредственной смежности для большего числа пар? Среди других мер близости можно отметить максимальное расстояние между элементами пар, являющихся смежными, среднее расстояние по таким ларам и т. д.

6.6. Измените алгоритмы 6.4а и 6.4 таким образом, чтобы все дочерние вершины некоторой вершины графа, являющиеся одинаковыми, не включались в тетрарное дерево

6.7. Определите в соответствии с системой, приведенной на рис. 6 4, метку (адрес) пиксела, координаты которого x_i и y_i принимают значения от 0 до 2^n-1.

68 Основные затраты на осуществление операций расщепления и слияния в алгоритме 6.6 связаны с необходимостью проверки однородности получаемых квадратов. Воспользуйтесь анализом, проведенным в разд. 4.4, чтобы показать, что затраты на операцию слияния не могут оказаться больше затрат на операцию расщепления.

6.9. Измените алгоритм 6.6 таким образом, чтобы он обеспечивал обнаружение границ всех областей двухуровневого изображения.

Приложение 6.А.

ЭЛЕМЕНТАРНЫЕ СВЕДЕНИЯ О ГРАФАХ

Теория графов имеет продолжительную историю, обширную литературу и многочисленные приложения Очевидно, знакомство с некоторыми элементарньми понятиями теории графов может очень существенно способствовать пониманию большинства тех алгоритмов обработки изображений, в которых используется аппарат теории графов. Целью данного приложения является изложение некоторых наиболее общих понятий теории графов

Граф представляет собой множество точек, или вершин, связанных между собой линиями, или ребрами. Примеры графов приведены на рис. 6.10, 6.11,*б*—

6.13. Степень вершины графа равна числу ребер, инцидентных этой вершине. Так, все вершины графа, приведенного на рис. 6.12, имеют четвертую степень, а граф, приведенный на рис. 6.13, имеет вершины различных степеней — второй и третьей. Улицы города являются примером графа — его вершины соответствуют перекресткам и тупикам, а ребра представляют части улиц, связывающие перекрестки или ведущие в тупики. В подобном «графе улиц» большая часть вершин имеет четвертую степень, а тупиковые улицы заканчиваются вершинами первой степени.

Ориентированным называется граф, некоторым или всем ребрам которого приписана определенная ориентация. В нашем примере с улицами города граф превращается в ориентированный, если учитываются только улицы с односторонним движением. В таких графах различают степень исхода и степень захода вершины, которые определяются как число дуг, исходящих из вершины и входящих в нее соответственно. Модификации этих терминов были предложены нами в разд. 6 7.

Маршрутом, связывающим вершину *А с* вершиной *В,* называется последовательность ребер, которую необходимо пройти, продвигаясь от вершины /1 ч вершине *В.* Маршрут, начальная и конечная вершины которого совпадают, называется **контуром.** Граф называется **связным,** если для любой пары его вершин существует маршрут.

Деревом называется связный граф, не содержащий контуров.

Граф, содержащий некоторые или все вершины и некоторые или все ребра графа *G* и не содержащий никаких иных вершин и ребер, называется **подграфом** указанного графа G. Подграф связного графа G, содержащий все вершины графа G и те его ребра, которые необходимы для того, чтобы данный подграф являлся связным и не содержал контуров, называется **остовным деревом** графа G.

Общепринятый способ изображения деревьев заключается в выборе вершины в качестве корня дерева, последующем размещении под ней всех соединенных с первоначально выделенной вершин и т. д. (см. рис 6.3). Все вершины, связанные с расположенной над ними вершиной *А,* называются ее дочерними вершинами (непосредственными **потомками),** а вершина *А* называется их **вершиной-предком** (родительской вершиной) [1].

[1] Следует отметить, что данное приложение написано весьма неформально и приводимые автором определения и терминология несколько отличаются от принятых в отечественной литературе. Частично (там, где это не требовало **полной** переделки текста) эти отклонения были устранены. — ***Прим. перев.***

Глава 7 ДВУХУРОВНЕВЫЕ ИЗОБРАЖЕНИЯ

7.1. ВВЕДЕНИЕ

Эта и две следующие главы посвящены изображениям класса 2, у которых хорошо определены области фиксированного серого тона или цвета. Основная проблема, возникающая при изучении таких изображений, связана с *формой* — понятием, которое не просто определить в количественных категориях. Эта проблема возникает при дискретизации аналогового двухуровневого изображения. Размер ячеек сетки выборки должен быть достаточно малым, чтобы очертания областей определенного цвета оставались

[1] Следует отметить, что данное приложение написано весьма неформально и приводимые автором определения и терминология несколько отличаются от принятых в отечественной литературе. Частично (там, где это не требовало **полной** переделки текста) эти отклонения были устранены. — ***Прим. перев.***

неизменными при восстановлении изображения. В разд. 7.2, 7.4 и 7.6 рассматривается этот, а также другие аспекты проблемы дискретизации. Еще одна группа задач, возникающих при обработке изображений этого типа, связана с их преобразованием в некоторое множество кривых и переходом от множества кривых к областям на плоскости. В разд. 7.5 обсуждается построение (выделение) контура области. Восстановление области по ее контуру — *заполнение* области — будет рассмотрено в гл. 8. Вместо того чтобы выделять контур области, можно применить к ней процедуру прореживания, обеспечивающую построение остова области — фигуру типа человечка из спичек, кривые и сегменты которого отражают форму изучаемой области. Алгоритмы *прореживания* рассматриваются в гл. 9.

Алгоритмы построения контура, заполнения контура и прореживания используются при решении различных задач, однако все эти алгоритмы имеют много общих свойств. Все они предусматривают обход области на плоскости, и, несмотря на очевидную простоту, эта процедура (осуществляемая, например, с помощью алгоритма 6.2) порождает целый ряд достаточно тонких задач. Нетрудно дать точные определения для построения контура, заполнения контура и прореживания плоской области в непрерывном случае при условии, что речь идет об ограниченных множествах, контуры которых удовлетворяют некоторым условиям гладкости. Труднее дать точные определения для дискретной плоскости. Геометрии дискретной плоскости посвящен разд. 7.3. К этой теме мы вернемся в разд. 7.6 и 7.7. И наконец, поскольку анализ формы области является основной целью построения контуров при работе с изображениями класса 2, мы посвящаем разд. 7.8 краткому обсуждению проблемы получения описаний формы области по данным, характеризующим ее контур.

7.2. ДИСКРЕТИЗАЦИЯ И ТОПОЛОГИЯ

Дискретизация двухуровневых или в более общем случае изображений класса 2 отличается особой спецификой, так как в состав этих изображений входят ступенчатые функции. Фурье-образы таких функций имеют ненулевые значения на всех частотах. Следовательно, не существует конечного интервала выборки, обеспечивающего отсутствие ошибки дискретизации. Интуитивно это кажется очевидным, если требуется сохранить информацию о точном расположении границ между областями. Одномерный вариант этой задачи проиллюстрирован на рис. 7.1. Если ошибка в определении положения границ считается допустимой, то решение может быть получено следующим образом.

Благодаря особенностям передаточной функции точки $g(r)$ (см. подразд. 2.3.2) выборочные значения не должны быть равны обязательно одному из двух исходных уровней серого тона. Вблизи границы, разделяющей уровни, они будут принимать промежуточные значения. Чтобы восстановить двухуровневое представле-

ние, выборочные значения следует сравнить с пороговым значением, лежащим между значениями исходных уровней серого тона. Ошибка, вносимая в результате этой операции в определение положения границы, не превышает половины длины интервала, в котором функция $g(r)$ принимает ненулевые значения. Предполагается, что этот интервал очень мал по сравнению с интервалом выборки, как и должно быть в любом правильно спроектированном устройстве дискретизации изображений.

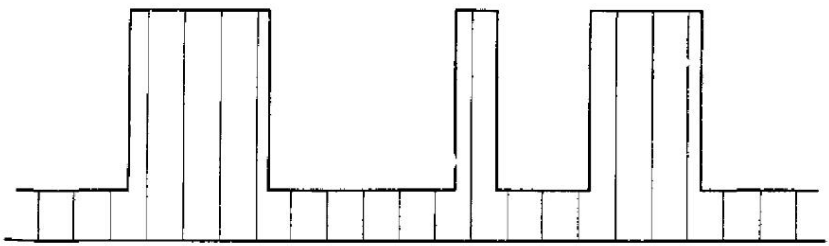


Рис 7.1. Дискретизация двухуровневого сигнала В дискретном представлении сигнала все пики будут сохранены, если расстояние между двумя выборочными значениями меньше минимальной ширины этих пиков

Пусть h — максимально допустимая ошибка. Заданная точность достигается, если интервал выборки меньше значения h. Ни один интервал не будет потерян, если минимальная ширина превышает значение h. Можно показать, что в случае, когда значение двузначной функции, представляющей двухуровневое изображение, изменяется только в тех точках, которые отстоят друг от Друга на расстояние, равное целому кратному значению h, фурье-образ этой функции будет на высоких частотах принимать нулевые значения, и, следовательно, возможна корректная дискретизация этой функции. Даже если последнее допущение нереалистично, реалистично допущение о минимальной ширине интервала, и чем меньше значение h относительно последней, тем меньше ошибка, вносимая дискретизацией. В частности, появляется возможность избежать возникновения *искажений* изображения вследствие недостаточно высокой частоты выборки, которые в случае двухуровневых изображений проявляются как потеря разрывов, и в результате функция представляется на некотором интервале однозначной, хотя в действительности она таковой не является.

Другой подход к дискретизации двухуровневых сигналов предусматривает их преобразование в многоградационные сигналы. Фильтр, осуществляющий замену значения яркости изображения в некоторой точке средним значением яркостей соседних с ней точек, является фильтром нижних частот — в результате фурье-образ функции, представляющей изображение, принимает на высоких частотах очень близкие к нулю значения. Следовательно, дискретизация полученного таким образом тонового изображения может

происходить без внесения каких-либо ошибок, за исключением возникающих в результате усреднения. При работе с изображениями этим подходом не всегда можно пользоваться, поскольку встречаются прикладные задачи (например, фотонабор), в которых выходной сигнал обязательно должен быть двухуровневым. Если выборочные значения разделяются по порогу таким образом, что получаемый в результате сигнал является двухуровневым, то в определение положения границ вносится ошибка, и, следовательно, данный метод оказывается эквивалентным первому.

Перенос задачи дискретизации (как и ее решения) с одномерного на двухмерный случай не тривиален. Если линию можно дискретизировать лишь одним способом (путем деления ее на интервалы), то в двухмерном случае возможности безграничны! Приходится задавать не только размеры сетки выборки, но и ее вид. По причинам, изложенным в подразд. 2.3.2, наше обсуждение будет вестись исключительно применительно к сеткам с квадратными ячейками. Если длина стороны квадратной ячейки равна h, то очевидно, что ни одна область не будет потеряна при дискретизации, если каждая из них достаточно велика для того, чтобы в наименьшую область можно было вписать квадрат сетки выборки. Это условие справедливо независимо от того, соответствует выборочное значение центру ячейки или оно определяется как среднее по всей ее площади. К сожалению, результаты этого способа дискретизации чувствительны как относительно переноса, так и ориентации области, что иллюстрируется на рис. 7.2. Более того, способ

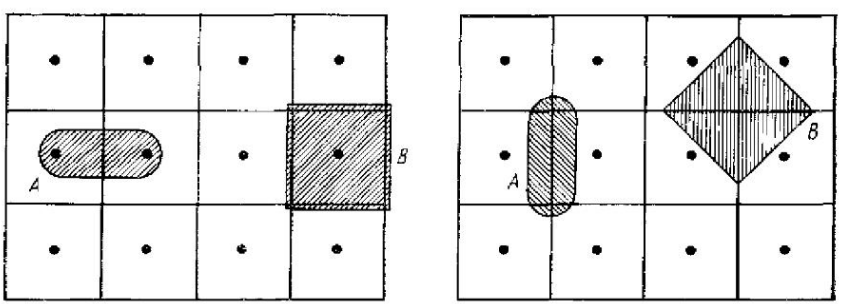


Рис. 7.2. Примеры трудностей, возникающих при задании критериев дискретизации изображений; при дискретизации левого рисунка области A и B поддаются обнаружению, а при дискретизации правого рисунка они пропускаются

никак не учитывает необходимость сохранения топологии областей при дискретизации. Ни одна из этих задач не возникала в одномерном случае!

С практической точки зрения главной проблемой при дискретизации изображений класса 2 является сохранение формы областей. Для обеспечения последнего необходимо, чтобы связные области непрерывного изображения оставались связными на ди-

скретном изображении, как и выполнение некоторых других условий. Необходимость сохранения связности автоматически порождает проблемы, относящиеся к топологии[1], и именно здесь начинаются трудности. Дело в том, что в сущности невозможно дать естественное определение топологии на множестве, элементы которого (в некотором смысле) изолированы друг от друга.

Однако существует выход. Вместо того чтобы заниматься преобразованием непрерывного изображения в некоторое множество дискретных пикселов, обратимся к преобразованию непрерывного изображения при помощи его восстановления, т. е. будем рассматривать непрерывные изображения, получаемые в результате заполнения элементов воспроизведения изображения цветом, в который окрашен расположенный в этом элементе пиксел. В таком случае нам придется иметь дело с взаимосвязями в парах таких изображений, где оба изображения являются непрерывными. В результате большая часть трудностей устраняется. Практическая реализация этого подхода дает неплохие результаты. Когда говорят о дискретных изображениях, обычно имеют в виду восстановленные изображения. Пусть I — исходное изображение. Каждому пикселу в процессе дискретизации присваивается тот цвет, в который окрашен центр соответствующей ячейки сетки выборки. В таком случае новое изображение I можно получить, заполняя каждый элемент (ячейку) воспроизведения изображения тем цветом, в который окрашен соответствующий пиксел. Этот процесс иллюстрируется рис. 7 3.

Прежде чем перейти к обсуждению топологических отношений, существующих между двумя изображениями, необходимо придать

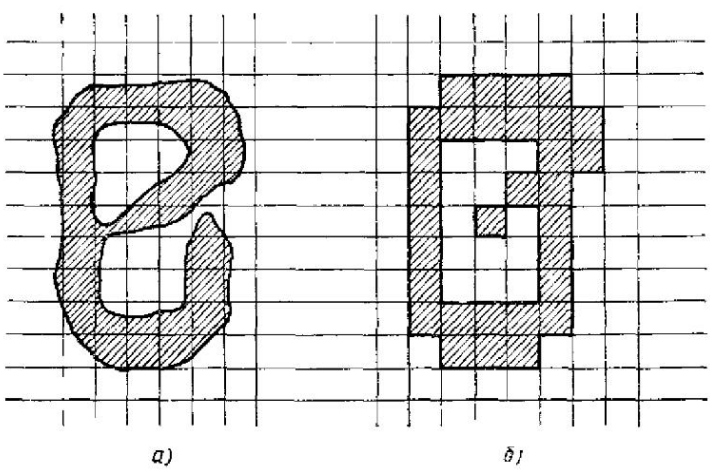


Рис. 7.3. Исходное изображение (а) и восстановленное, полученное после дискретизации с помощью сетки выборки с квадратичными ячейками (б)

[1] Мы дадим краткое объяснение того, что понимается под *топологией* для читателей, не знакомых с этим понятием

ясность используемой терминологии. Говорят, что два множества A и B топологически эквивалентны, если между Л и В существует некоторое взаимно-однозначное отображение, причем как отображение A в B, так и обратное отображение из В в Л непрерывны ([7.4, p. 87]). Поскольку при обработке изображений обычно встречается топология, индуцированная евклидовой метрикой, то топологическая эквивалентность означает существование такого взаимно-однозначного отображения, при котором точки, расположенные вблизи друг от друга, отображаются в точки, расположенные вблизи друг от друга[1], и наоборот. Содержательно это означает, что одно множество можно отобразить на другое при помощи растяжения и сжатия, не допуская при этом никаких разрывов и разрезов. Следовательно, область, в которой имеется дыра, не может быть топологически эквивалентна области, не имеющей дыры, поскольку соответствующее преобразование потребовало бы проведения разреза.

Топологические понятия нетрудно определить на любом из приведенных на рис. 7.3 изображений: оба расположены на непрерывной плоскости, на которой строго определена евклидова метрика. Кроме того, можно показать, что эти изображения действительно топологически эквивалентны. Строгое доказательство этого факта увело бы нас далеко за рамки, определенные нами для данной книги, но и без обращения какому бы то ни было формализму достаточно очевидно, что, например, оба заштрихованных множества являются связными и что в каждом из них имеется одно отверстие. Этот пример показывает также, что сохранение топологии может служить необходимым, но не достаточным условием для сохранения формы. Результат преобразования изображения, приведенный на рис. 7.3,*б*, явно неприемлем с этой точки зрения. Несмотря на то, что требование сохранения числа связных областей каждого цвета — условие недостаточное, это, тем не менее, первое условие, которое должно быть выполнено.

В связи с простотой критерия дискретизации для одномерного случая может возникнуть искушение потребовать выполнения этого условия на всех линиях развертки. К сожалению, этот критерий может не работать на строках, близких к касательным границ областей. Эту проблему надо разрешать непосредственно, чем мы и займемся в разд. 7 4. Сначала нам придется сделать небольшое отступление, чтобы ввести ряд простых понятий геометрии дискретной плоскости.

7.3. ЭЛЕМЕНТЫ ГЕОМЕТРИИ ДИСКРЕТНОЙ ПЛОСКОСТИ

Как уже отмечалось во введении, многие геометрические понятия, которые точно определены для непрерывных изображений,

[1] Читателям, считающим термин «вблизи» недостаточно точным, следует обратиться к какому-либо учебнику по математическому анализу и посмотреть определение непрерывной функции.

не имеют аналогов для дискретных изображений (множеств пикселов). Итак, необходимо уделить некоторое внимание определению соответствующих понятий для дискретных изображений.

Определение 7.1. Два пиксела называются *непосредственными соседями* (н-соседями), если соответствующие элементы имеют общую сторону, и *косвенными соседями* (к-соеедяоди), если соответствующие элементы касаются лишь углами. Термин *сосед* относится к соседству обоих типов. Термин *N-сосед,* где O<=N<=7, будет использоваться для обозначения пиксела, позиция которого определяется его положением на рис. 7.4 в соответствии с различными значениями *N.*

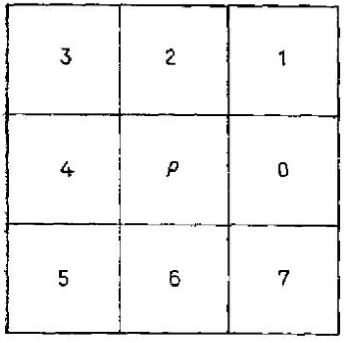


Рис. 7.4. Обозначение, определяющее положение отдельных пикселов относительно положения пиксела *P*

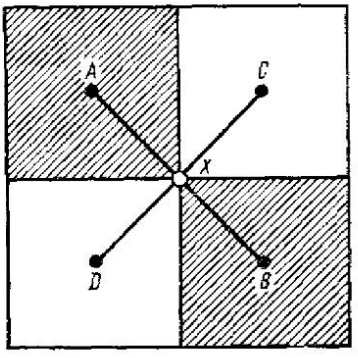


Рис. 7.5. Пример топологических противоречий, возникающих при определении связности на дискретной сетке

Отметим, что н-соседи являются N-соседями при четных значениях *N,* а к-соседи соответствуют нечетным значениям *N.*

Определение 7.2. *К-маршрутом* (или просто *маршрутом)* называется такая последовательность пикселов $A_1, A_2, ..., A_n$, в которой при /г>1 пиксел A_{k-1} является соседом пиксела A_k и при $k<n$ пиксел A_{k+1} является соседом пиксела A_k. Если на аналогичную последовательность наложить условие, заключающееся в том, чтобы входящие в нее пикселы являлись не просто соседями, а н-соседями, то такую последовательность будем называть н-маршрутом. *Простым* называется маршрут, все пикселы которого являются различными и ни один из них не имеет в маршруте более н-соседей. *Замкнутым* называется маршрут, первый и последний пикселы которого совпадают.

Определение 7.3. Множество пикселов S называется *связным* (или *к-связным),* если для каждой пары пикселов *C* и *D* в 5 существует к-маршрут, начальным и конечным элементами которого служат пикселы C и D соответственно, а все его остальные пикселы принадлежат множеству 5. Значения термина «н-связное» очевидно.

В литературе проблеме связности уделено определенное внимание, поскольку охарактеризовать ее можно двумя различными способами. Рассмотрим парадокс, который иллюстрируется рис. 7.5.

Естественно, было бы желательно распространить понятие связности, введенное для дискретной плоскости, на плоскость непрерывную. Если множество пикселов является связным в соответствии с введенным выше определением, то желательно, чтобы связным являлось и множество элементов воспроизведения изображения. Рассматривая пример, приведенный на рис. 7.5, естественно допустить, что на непрерывной плоскости множества темных и светлых точек не пересекаются и оба эти множества полностью покрывают изображенную часть непрерывной плоскости, поскольку отсутствуют точки какого-либо другого цвета. Если связность определяется как к-связность, то возникает следующая ситуация: маршрут, соединяющий пиксел *A* с пикселом *B* (см. рис. 7.5), может пересекать маршрут, соединяющий пиксел *C* с пикселом *D,* хотя каждый из этих двух маршрутов целиком входит в одно (соответствующее) из двух непересекающихся множеств. Таким образом, точка *X* должна одновременно принадлежать как множеству светлых, так и множеству темных точек. Если связность определяется как н-связность, то ни множество темных, ни множество светлых пикселов не являются связными. Отсюда следует, что маршрут, связывающий пиксел *A* с пикселом *B* (или пиксел С с пикселом *D),* не может целиком принадлежать лишь одному из множеств. Поскольку отрезки прямых *AB* и *CD* пересекаются в точке *X,* эта точка не должна принадлежать ни одному из множеств. Однако последнее противоречит допущению, согласно которому темные и светлые элементы полностью покрывают область плоскости, изображенную на рис. 7.5.

Приведенный парадокс своим возникновением обязан небрежности, допущенной при установлении соответствия между множествами пикселов и множествами элементов воспроизведения изображения. Множества, заданные на непрерывной плоскости, могут включать, но могут и не включать свои границы[1], а в нашем примере решение, касающееся включения границ, определяет характер связности. Имеется ряд возможностей — одна из них состоит в том, что темные элементы (согласно принятому допущению) включают свои границы (т. е. являются *замкнутыми* множествами в топологическом смысле), а светлые элементы (согласно принятому допущению) не включают свои границы (т. е. являются *открытыми* множествами). В таком случае множество темных элементов является связным, а множество светлых элементов таковым не является. Это становится совершенно очевидным, если

[1] Формально граница множества S определяется как множество всех точек плоскости, обладающих следующим свойством. Если точка *P* принадлежит границе множества S, то все окружности с центром в точке *P* включают как точки, принадлежащие S, так и его дополнению, независимо от того, насколько мал радиус таких окружностей. Точка *P* может принадлежать, но может и не принадлежать множеству S.

ввести (Систему координат x, y и задать области, представленные на рис. 7.5, аналитически. Если Z — длина стороны квадрата и начало координат находится в нижнем левом углу, то имеем следующее:

Квадрат A (темный) $0 <= x <= Z \quad Z <= y <= 2Z$.

Квадрат B (темный) $O <= y <= Z \quad Z <= x <= 2Z$.

Квадрат C (светлый) $Z < x < 2Z \quad Z < y < 2Z$.

Квадрат D (светлый) $0 < x < Z \quad 0 < y < Z$.

Точка X имеет координаты (Z, Z) и совершенно очевидно является темной.

Рассматривая некоторое изображение как множество пикселов, получаем непротиворечивые определения, используя понятие н-связности для светлых пикселов и к-связности — для темных. Подобный выбор многократно описан в литературе, однако он перестает быть оправданным при работе с изображениями, состоящими из пикселов нескольких цветов (более двух). Можно допускать, что включение в множество некоторых точек его границы определяется не цветом последних, а ориентацией этой границы. Перпендикуляр к границе некоторого объединения квадратных элементов может быть проведен извне по следующим направлениям: 0, 2, 4 и 6 (обозначения направлений соответствуют рис. 7.4). Можно считать что точки границы принадлежат этому множеству в том и только в том случае, если выполняется одно из двух следующих условий: а) перпендикуляр проведен по направлению 0 или 2; б) данная точка является углом, перпендикуляры к сторонам которого проведены по направлениям 0 и 2 соответственно. Эти условия проиллюстрированы на рис. 7.6. При этих условиях пикселы одного цвета (Соприкасающиеся углами, будут считаться связными в том и только в том случае, если направление биссектрис этих углов соответствует 1 (см. рис. 7.4). Согласно этому определению анали-

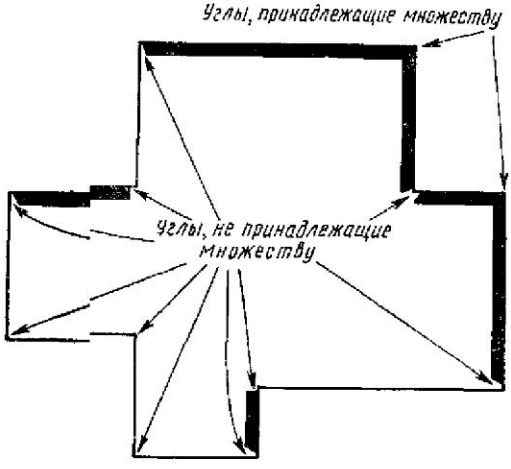


Рис. 7.6. Определение вхождения граничных точек в множество элементов воспроизведения изображения в зависимости от ориентации границы

Жирными линиями отмечены участки границы, которые состоят из точек, принадлежащих множеству элементов воспроизведения изображения

тическое описание некоторого прямоугольника будет иметь следующий вид:

$$a < x <= b \quad c < y <= d.$$

Практический выбор определения может зависеть от целого ряда факторов. Например, в растровых графических устройствах прямые строятся таким образом, что составляющие их пикселы соприкасаются только углами. Следовательно, чтобы определение связности было совместимо с данным способом реализации, необходимо для изображаемых объектов предусмотреть к-связность, а для фона — н-связность. Этот принцип остается справедливым даже при использовании цветных графических устройств, поскольку объекты и фон представляют собой такие элементы воспроизводимого отображения, которые поддаются точному определению. Единственное обстоятельство, которое потенциально может составить проблему, возникает при изменении задания изображаемых объектов и фона в процессе формирования отображения, в результате чего происходит пересечение линий разных цветов. Можно пользоваться также и более сложными определениями связности, однако их обсуждение выходит за рамки, установленные нами для данной книги (см. работу [3.7, p. 62—64]).

7.4. ТЕОРЕМА ОТСЧЕТОВ ДЛЯ ИЗОБРАЖЕНИЙ КЛАССА 2

Продолжим изучение процесса дискретизации изображений класса 2, осуществляемой с помощью сетки выборки с квадратными ячейками. Если ввести условие, требующее, чтобы каждая область определенного цвета являлась достаточно большой для того, чтобы она покрывала элемент дискретизации при произвольной ориентации последнего, то появляется гарантия, что ни одна область исходного изображения I не будет отсутствовать на восстановленном изображении J, однако и в этом случае могут возникать существенные искажения формы областей. Определение 7.4 вводит более сильное условие; мы докажем, что его выполнение обеспечивает сохранение формы.

Определение 7.4. Изображение класса 2 и сетка выборки с квадратными элементами называются совместимыми при выполнении следующих двух условий: а) существует такое число $d > \text{sqrt}(2)\, h$, где h — длина стороны элемента сетки выборки, что для каждой граничной точки каждой области R заданного цвета найдется окружность C диаметром d, касающаяся границы и полностью лежащая внутри области R; б) то же самое справедливо и для дополнения области R.

Это условие совместимости задает нижние границы для ширины всех областей и кривизны их контуров. Так, в частности, не допускается наличие углов. Последнее может представляться серьезным ограничением, однако на самом деле это не так. Действительно, допустим, имеется класс объектов, контуры которых со-

держат углы. Выберем радиус кривизны r и заменим каждый угол дугой окружности радиуса r. Если r достаточно мал, то может не возникнуть заметных искажений формы объектов, но теперь появилась возможность найти совместимую сетку выборки. Достоинством определения 7.4 является то, что при решении ряда прикладных задач оказывается возможной проверка выполнения условия совместимости. Так, например, литеры типографского шрифта часто имеют такие начертания, что оказывается возможным измерить их минимальную ширину и минимальную кривизну.

Выполнение условия совместимости гарантирует наличие ряда свойств, которые характеризуются приведенными ниже леммами. Термин «сосед» используется в смысле определения 7.1, т. е. соседями считаются элементы, соприкасающиеся как углами, так и сторонами.

Лемма 7.1. Если условие совместимости выполняется и элемент дискретизации Z пересекается с областью R, то либо центр элемента дискретизации Z расположен в области R, либо элемент дискретизации Z имеет соседа, центр которого расположен в области R.

Доказательство. Пусть C — одна из окружностей диаметром d, расположенная в области R и пересекающаяся с элементом дискретизации Z, но не содержащая его центр. Тогда окружность C должна касаться прямой X, *как* показано на рис. 7.7,с. Если ок-

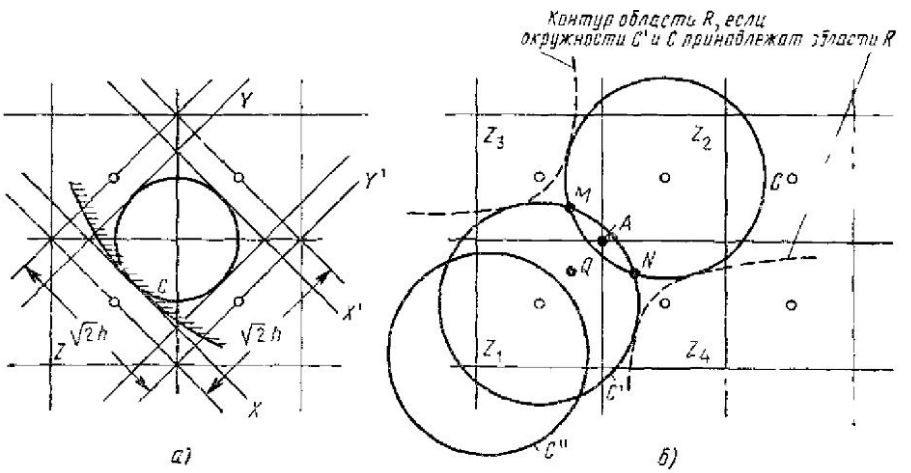


Рис. 7 7 Иллюстрации к доказательствам леммы 7.1 (а) и леммы 7.2 *(б)*

ружность C не содержит центров каких-либо других элементов дискретизации, то она должна также касаться прямых X', Y и Y'. Однако поскольку расстояние между любыми двумя такими параллельными прямыми меньше значения sqrt(2) h, то, следовательно, оно меньше значения d. В результате мы пришли к противоречию!

Лемма 7.2. Если условие совместимости выполняется и область R содержит центры двух элементов дискретизации Z_1 и Z_2, являющихся соседями, то эти элементы дискретизации Z_1 и Z_2 имеют либо общую сторону, либо некоторого соседа с центром, расположенным в области R, причем этот сосед имеет общую сторону с каждым из элементов дискретизации Z_1 и Z_2.

Доказательство. Допустим, что указанные элементы дискретизации не имеют ни одной общей стороны и центры элементов дискретизации Z_3 и Z_4 расположены вне области R (рис. *7.7,б)*. Если бы угол A находился в области R, то он должен был бы быть расположен внутри окружности диаметром d, целиком лежащей в области R. Окружность C представляет собой такую окружность, которая не содержит центров элементов дискретизации Z_3 и Z_4. Рассмотрим теперь аналогичную окружность, содержащую центр элемента дискретизации Z_1. Последняя может пересекаться с окружностью C (вариант расположения C') или не пересекаться с ней (вариант расположения C''). Если пересечение имеет место, то пусть M и N — соответствующая пара точек пересечения. Обе эти точки расположены в области R, и любая окружность диаметром d, содержащая их, должна содержать также центр одного из элементов дискретизации Z_3 и Z_4. В результате мы пришли к противоречию. Если окружности не пересекаются, то, как показано на рис. 7.7,*б*, должна существовать точна Q, не принадлежащая области R и принадлежащая элементу дискретизации Z\. Таким образом, невозможно построить окружность диаметром d, содержащую указанную точку Q и не содержащую центры элементов дискретизации Z_1 и Z_2. Это означает, что условие совместимости не выполняется. Следовательно, лемма для случая, когда угол A расположен в области $(R$, доказана. Аналогичным образом доказывается лемма для случая, когда угол A расположен вне области R.

Следствие. Выполнение условий совместимости означает, что все связные области дискретного изображения являются непосредственно связными.

Исключение касания углами не является особенно удивительным фактом. Оно было основным источником неопределенности в установлении связности и потому должно быть запрещено с помощью некоторого набора условий, обеспечивающих, помимо прочего, сохранение связности.

Утверждение 7.1. Пусть A и B — две точки множества R, принадлежащего непрерывному изображению I, и K — кривая, соединяющая их и целиком лежащая в множестве R. Последнее отображается в множестве R', принадлежащем восстановленному изображению J. В таком случае все точки кривой K можно отобразить на кривую K', целиком лежащую в множестве R', причем ни для одной пары соответствующих точек кривых K и K' расстояние, разделяющее их, не превышает значения d.

Доказательство. Если допустить, что условие совместимости выполняется, то из этого следует, что каждая точка кривой K расположена внутри окружности диаметром d, целиком лежащей в

•множестве R. Размеры этой окружности таковы, что в ней должен быть расположен некоторый элемент дискретизации. Следовательно, можно установить некоторое соответствие между всеми точками кривой K и центрами элементов дискретизации (рис. 7.8,*а*). Согласно лемме 7.2 существует некоторый маршрут, соединяющий центры всех этих элементов дискретизации, причем он

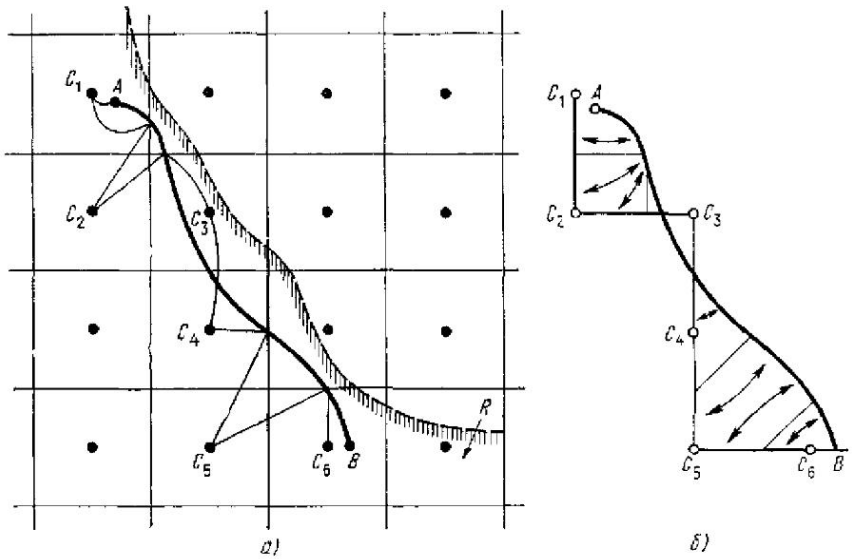


Рис. 7.8. Иллюстрации к доказательству утверждения 7.1

будет пересекать только стороны элементов дискретизации, не проходя через их углы. Части маршрута, лежащей в пределах каждого элемента дискретизации, можно поставить в соответствие часть кривой K, отображенную на этот элемент (рис. 7.8,*б*). Очевидно, можно найти такое отображение точек кривой K на данный маршрут, что расстояние между соответствующими точками отображения не будет превышать d.

Следствие. Границы соответствующих областей изображений I и J при наложении этих изображений разделены расстоянием, не превышающим d.

Этот результат можно интерпретировать как утверждение о соответствии (сохранении) формы непрерывного изображения и изображения, восстановленного по дискретизированному варианту исходного изображения. Однако могут возникнуть опасения относительно возможности разрыва или слияния некоторых областей. Теперь покажем, что этого не происходит и изображения I, J топологически эквивалентны.

Теорема 7.1. Выполнение условия совместимости обеспечивает сохранение топологии.

Доказательство. Определим некоторое взаимно-однозначное отображение для множества и его дискретного аналога, причем это отображение является непрерывным в обоих направлениях. •Если элемент дискретизации полностью покрывается областью, то соответствующее отображение должно быть тождественным. В противном случае необходимо рассмотреть варианты избытка и недостатка. В примере, приведенном на рис. 7.9, треугольная область

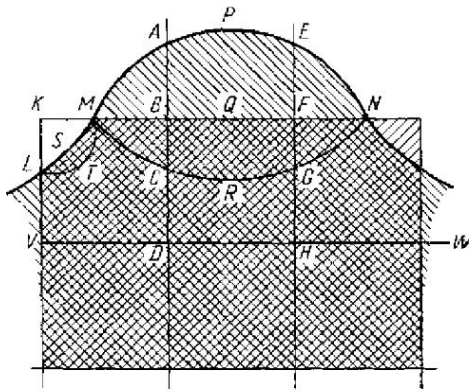


Рис. 7.9. Иллюстрация к доказательству теоремы 7.1

Исходная область заштрихована с наклоном влево, а восстановленная — вправо. Отрезки прямых *BD* и *FH* дискретного изображения отображены на отрезки прямых *CD* и *GH* непрерывного изображения, отрезки прямых *AB* и *EF* — на отрезки *BC* и *FG*, отрезок *BQF* — на *CRG*, *APE* — *на BQF* и т д, точки *L*, *M* и *N* — фиксированы; дуга *LTM* отображена на *LSM*, а *LSM* — на (кусочно-линейную) дугу *LKM*

KLM представляет собой недостаток, а именно, ту часть элемента дискретизации, которая не входит в непрерывную область. Область *MAPENFQBM* представляет собой избыток, поскольку является частью непрерывной области, лежащей за пределами восстановленной области. Если элемент дискретизации содержит такую часть области или является смежным с ней, то он подвергается разбиению на части. Части элемента, не входящие ни в одну из таких областей, рассматриваются как точки непрерывной области, в которой все точки элемента дискретизации имеют отображение. (Так, элемент *DBFH* стягивается в *DCRGH*, часть элемента *VLSMBD* — в *VLTMCD* и т. д.) Остальные части отображаются Б соответствующие симметричные (относительно непрерывной границы) области, как показано на рис. 7.9. Наши две леммы гарантируют непрерывность данных преобразований. В частности, из леммы 7.1 следует, что любой избыток оказывается смежным с элементом дискретизации того же цвета. Поскольку эта лемма справедлива как для множества, так и для его дополнения, из нее следует, что заполнение недостатков не приводит к слиянию областей. Лемма 7.2 гарантирует отсутствие соприкосновений углов и, следовательно, возможность выполнения всех преобразований вдоль сторон элементов, как показано на рис. 7.9.

Эта теорема вместе с утверждением 7.1 указывает, что восстановленный вариант области топологически эквивалентен исходной непрерывной области и каждую кривую, принадлежащую области, можно отобразить в кривую другой области таким образом, что расстояние между соответствующими точками этих кривых

сравнимо по порядку с размерами элемента сетки дискретизации. Итак, можно с удовлетворением констатировать, что дискретизация, при которой выполняется условие совместимости, обеспечивает сохранение формы.

В топологическом смысле условие совместимости означает, что все непрерывные множества являются открытыми, и оно гарантирует, что восстановленные множества также являются открытыми. Этот подход оставляет, судя по всему, открытой проблему тонких линий, которые теоретически имеют нулевую толщину, а топологически не являются открытыми множествами. Отметим, что в реальных условиях линии всегда имеют конечную толщину и их невозможно корректно дискретизировать, если условие совместимости для них не выполняется. С другой стороны, может быть целесообразно как с теоретической, так и с практической точек зрения рассматривать отдельно восстановление таких линий и множеств, имеющих конечную толщину. Мы так и поступим в разд. 7.6.

7.5. ПОСТРОЕНИЕ КОНТУРА

Понятие границы множества, расположенного на непрерывной плоскости, вполне очевидно. Такую границу образует множество всех точек, которые обладают следующим свойством: независимо от того, сколь мала выбранная окрестность этих точек, она содержит точки, лежащие как внутри множества, так и вне его. Не столь очевидно соответствующее понятие для дискретной плоскости. Дадим несколько определений, которые позволят прояснить это понятие. Отметим также, что термин «контур» мы сохраним для случая дискретной плоскости, а термином граница будем пользоваться лишь применительно к множествам, расположенным на непрерывной плоскости.

Определение 7.5 *Контуром,* или *к-контуром,* связного множества пикселов R будем называть множество всех пикселов из R, каждый из которых имеет, по меньшей мере, одного н-соседа, расположенного вне R; *н-контуром* множества R будем называть множество всех пикселов из R, каждый из которых имеет, по меньшей мере, одного соседа, расположенного вне R.

7.5.1. ПОСТРОЕНИЕ ОДИНОЧНОГО КОНТУРА

Обход пикселов контура можно осуществлять в соответствии с маршрутом, причем для такого обхода всегда можно использовать некоторый замкнутый маршрут Далее, имея дело с контурами, всегда будем иметь в виду вполне определенную процедуру обхода, а именно процедуру, задаваемую алгоритмом 7 1. В ней используются понятие N-соседа (см. определение 7.1) и обозначения, введенные с помощью рис. 7.4. Все численные операции с N выполняются по модулю 8. Данный алгоритм можно описать в терминах поведения наблюдателя, движущегося вдоль пикселов, принадлежащих множеству, и выбирающего крайний правый пик-

сел. Начальный пиксел A может определяться рядом способов, в том числе с помощью обхода плоскости сверху вниз и слева направо. Процедура построения контура заканчивается, когда очередным пикселом, осматриваемым в процессе обхода, оказывается начальный. Поскольку такое же положение имеет место в начале обхода, флаг *первый* используется для того, чтобы можно было различать начало алгоритма и возвращение в начальный пиксел. Цикл, предусмотренный на шагах алгоритма 5—9, выполняется не более трех раз, чтобы избежать хождения по кругу вокруг множества, содержащего лишь один пиксел.

Алгоритм 7.1. Алгоритм построения контура

Процедура *TRACER*

Обозначения. A — начальная точка контура множества R; C — текущая точка, окрестность которой в данный момент исследуется; S — направление поиска, выраженное в коде, приведенном на рис. 7.4; *первый* — флаг, имеющий значение «истина» только в начале построения; *обнаружен* — флаг, имеющий значение «истина» только при обнаружении очередной точки контура.

0. В контуре выбирается точка A таким образом, чтобы ее 4-со сед не принадлежал рассматриваемому множеству.
1. Точке A присваивается значение текущей точки C, направле нию поиска S — значение 6 и флагу *первый* — значение *истина.*
2. **While** C отличается от A или флаг *первый* имеет значение *ис тина* **do** шаги 3—10.
   Begin.
3. Флагу *обнаружен* присваивается значение *ложь.*
4. **While** значение флага *обнаружен* равно *ложь* **do** шаги 5—9, но не более трех раз.
   Begin.
5. If B, являющаяся (S—1)-соседом C, принадлежит множеству R, then
   Begin.
6. Точке B присваивается значение текущей точ ки С, направлению поиска 5 — значение S—2 и флагу *обнаружен* — значение *истина.*
   End.
7. **Else if** ***B,*** являющийся S-соседом С, принадлежит множеству R, **then** точке B присваивается значение текущей точки C и флагу *обнаружен* — значение *истина.*
8. **Else if** B, являющаяся (S+1)-соседом С, принад лежит множеству R, **then** точке B присваивается значение текущей точки С и флагу *обнаружен* — значение *истина.*
9. Else значение направления поиска S увеличивает ся на 2.
   End.

10. Флагу *первый* присваивается значение *ложь*.
End.
11. **Конец алгоритма**.

Данный алгоритм предусматривает по одному проходу на каждое отверстие области и, кроме того, один проход — на построение внешнего контура области. Следовательно, он должен применяться в сочетании с каким-либо алгоритмом поиска, позволяющим обнаруживать отверстия, расположенные внутри области. Кратко опишем один из таких алгоритмов. Обход образует замкнутый к-хмаршрут и предусматривает просмотр внешних контуров против часовой стрелки, а контуров отверстий — по часовой стрелке. Если в результате работы алгоритма должно порождаться некоторое описание контура, то можно воспользоваться направлениями поиска и координатами *x, y* точки *A*. Последние выводятся в первую очередь, а затем каждый раз, когда некоторому пикселу присваивается значение *текущий (С),* выводится значение направления поиска S. В результате порождается описание контура, представленное в цепном коде (см. подразд. 1.2.3). Описания, представленные в цепном коде, могут также храниться в оперативной памяти и использоваться при обходе внутренних частей области *в* процессе поиска отверстий.

7.5.2. ОБХОД ВСЕХ КОНТУРОВ ОБЛАСТИ

Алгоритм 7.2 определяет все контуры области с помощью «писанной выше процедуры *TRACER* (ОБНАРУЖИТЕЛЬ КОНТУРА) и предусматривает, что контуры, обнаруженные с помощью этой процедуры, помещаются в очередь Q. Поскольку описания контуров включают как пары значений координат *x, y,* так и кодовые последовательности, записанные в цепном коде, при организации очереди Q следует проявить определенную осторожность. Чтобы упростить описание алгоритма, допустим, что очередь Q содержит координаты *x, y* пиксела *P* и значение цепного кода *c*. Если пиксел является начальной точкой контура, то *c=8*. Практически достаточно иметь массив, в котором хранятся только значения цепного кода, и использовать символ 8 для обращения к отдельному массиву, содержащему начальные точки. В таком случае координаты *x, y* каждой новой точки можно, учитывая информацию, содержащуюся в цепном коде, получать исходя из координат предыдущей точки. Данный алгоритм предусматривает также введение на изображении соответствующей маркировки для пикселов, выделенных в качестве элементов контура. Предполагается, что при работе с двухуровневыми изображениями первоначально пикселы, входящие в область, помечаются единицей, а не входящие в нее, — нулем. Процедуру *TRACER* можно модифицировать таким образом, что при присвоении пикселу метки «текущая точка» его значение увеличивается на 1. В результате в конце процесса построения контура образующие его пикселы будут иметь значения 2 или более. В частности, это значение будет рав-

но единице плюс число осмотров соответствующего пиксела в процессе обхода. Эти значения используются при поиске отверстий внутри области. Следующий метод позволяет избежать поиска в части изображения, лежащей за пределами представляющей интерес области.

После построения внешнего контура и введения его в очередь Q начинается изучение содержимого последней. Если встречается точка, расположенная на опускающейся дуге, поиск направляется вправо. Такие пикселы легко определить, указав, что предшествующий элемент цепного кода должен иметь значения от 4 до 7, а следующий — от 5 до 7. Число 4 включено в диапазон допустимых значений предыдущего элемента для того, чтобы учесть точки перегиба, как показано на рис. 7.10. Поскольку для первого элемента направление предыдущего элемента неизвестно до тех пор, пока не будет установлено значение последнего элемента цепного кода данного контура, проверка первого элемента как начальной точки должна быть отложена до конца процедуры. (Мы не включили эту специфическую часть в описание алгоритма 7.2 для его упрощения.) В процессе просмотра по горизонтали необходимо отыскивать либо начало какого-либо отверстия, либо другую сторону внешнего контура. Эта задача осложняется тем, что некоторые пикселы могут одновременно входить во внешний контур какого-либо отверстия. Если известно, что таких пикселов нет, то необходимо искать лишь последовательности, состоящие из двух пикселов и имеющие значение 01 (начало отверстия), и пары пикселов, имеющие значения 2 (или более) и 0 (внешний контур). При наличии пикселов, одновременно входящих в контур отверстия и внешний контур, требуется предусмотреть более сложные процедуры контроля, которые представлены как часть алгоритма 7 2.

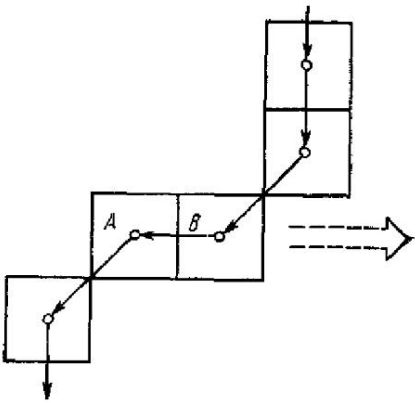


Рис. 7.10. Начало просмотра внутренней части области в точке перегиба внешнего контура. Цепной код, представляющий часть контура, изображенную с помощью стрелок, равен 665 456; точка *B* не выбирается в качестве начальной, а точка *A* выбирается

Алгоритм 7.2. Полный алгоритм построения контура.

Обозначения. В очереди Q содержатся адреса пикселов *P* и значения цепного кода *c*. Процедура *REMOVE* — та же, что и в алгоритме 6.3.

1. Построение внешнего контура при помощи обращения к процедуре *TRACER*. Обнаруженные точки и соответствующие цепные коды заносятся в очередь Q.

2. While Q — непустая do шаги 3—8.
   Begin.
3. *(P, c) = REMOVE (C).* If c равно 8 {*P* — начальная точка}, then c_0 приравнивается c и удаляется еще один пиксел: *(P,c)=REMOVE (C).*
4. If значения c_0 лежат в диапазоне 4—7 и значения c — в диапазоне 5—7, then do шаги 5—7.
   Begin.
5. Начиная с *P,* проводится поиск в *x*-направлении и осмотр троек последовательных пикселов *A, B, C.*
6. If *A* равно 0 и В равно 1, or if Л равно 0, *B* равно 2 и С равно 0, then *B* используется в качестве начальной точки и производится обращение к процедуре *TRACER,* что позволяет ввести контур отверстия в очередь Q. После вывода результатов goto шаг 8.
7. Else if *A* равно О, *B* больше 2 и Q равно 0, or if *A* равно 1, *B* равно 2 и Q равно 0, then goto шаг 8.
   End.
8. Установка $c_0 = c$.
   End.
9. Конец алгоритма.

Для обозначения начала непросмотренного отверстия используется код 020, поскольку некоторая дуга его контура может совпадать с некоторой дугой внешнего контура (рис. 7.11,*а*). С помощью кода 030 (или 040 и т. д.) обозначается просмотренное отверстие такого же типа, и после того, как поиск доходит до внешнего контура, он должен быть прекращен. Отметим, что области, целиком лежащие внутри отверстий (рис. 7.11,*б*), игнорируются. Дело в том, что контур отверстия, содержащего такую область, будет обнаружен в первую очередь (направление просмотра *UU* на рис. 7.11,*б*), после чего он будет построен и зарегистрирован. При перемещении направления просмотра вниз (например, направление *VV* на том же рисунке) просмотр прекратится вследствие обнаружения кодов 120 или 030. Поскольку контуры отверстий, как и внешние контуры, встреченные при просмотре изображения, служат сигналом к прекращению просмотра, контуры отверстий также следует использовать в качестве отправного элемента поиска. Это осуществляется автоматически, так как эти контуры также помещаются в очередь Q.

На рис. 7.12 показан порядок обхода внутренней части области. Вложенные области могут быть обнаружены при последующих просмотрах изображения. Для повышения эффективности такого поиска в очередь можно поместить координаты *x, y* тех пикселов, на которых закончился просмотр внутренней части области по горизонтальному направлению; эта очередь должна просматриваться после построения контуров каждой связной области.

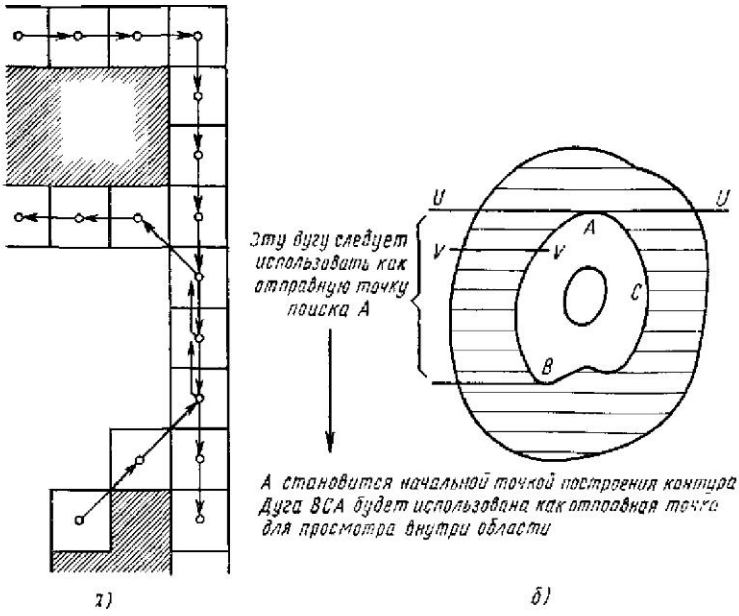


Рис. 7.11. Пример дуги, являющейся одновременно частью внешнего контура и частью контура отверстия (а), порядок просмотра вложенных областей (б)

Алгоритм 7.2 является эффективным, поскольку элементы изображения, не входящие ни в один из контуров, должны просматриваться лишь однократно, а элементы, принадлежащие одному из контуров, — лишь дважды. Так как нет необходимости помечать пикселы значениями, превышающими 3, алгоритм можно видоизменить таким образом, чтобы исключить приращение значений пикселов, после того как оно становится равным трем. В этом случае для хранения одного пиксела достаточно затрачивать два бита.

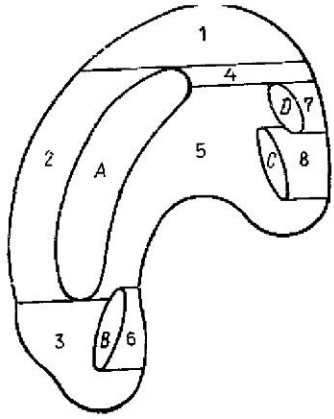


Рис. 7.12. Порядок просмотра внутренней части области

7.6. КРИВЫЕ И ЛИНИИ НА ДИСКРЕТНОЙ СЕТКЕ

Естественное геометрическое разделение кривых или линий (объектов, имеющих нулевую ширину) и областей на плоскости (объектов, имеющих конечную ширину) становится расплывчатым при переходе к заданию их на дискретной сетке. Если на такой сетке задано некоторое множество пикселов, то не всегда очевидно, является оно результатом дискретизации некоторой тонкой

кривой или некоторой целой области. Кроме того, евклидово определение прямой как кратчайшего расстояния между двумя точками теряет смысл при определении расстояния на сетке. В примере, приведенном на рис. 7.13, представлены два к-маршрута

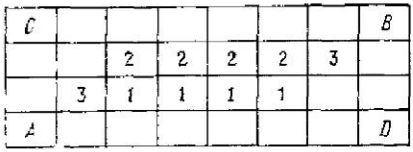


Рис. 7 13 Иллюстрация неединственности кратчайшего пути между двумя точками на дискретной сетке. Пикселы, образующие первый маршрут, обозначены единицами; пикселы, образующие второй маршрут, — двойками; пикселы, обозначенные тройками, принадлежат обоим маршрутам

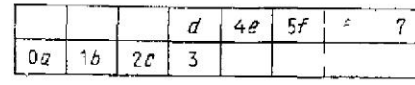


Рис. 7.14. Пикселы линии, связывающей точки 0 и 7, снабжены числовыми метками, а пикселы, образующие линию *Of(af),* помечены строчными буквами; пикселы, снабженные двумя метками, входят одновременно в обе линии

между точками *A* и *B,* каждый из которых состоит из шести пикселов. В данном случае утрачивается единственность евклидова определения. К тому же, две такие прямые могут пересекаться более чем в одном пикселе. Читатель может убедиться в этом, рассмотрев пересечение одной из линий, соединяющих точки *A и B, с* аналогичной линией, соединяющей точки C и *D.* В зависимости от выбора пары пересекающихся линий пересечение будет включать от нуля до четырех пикселов! Это осложнение не исчезнет, если собственно пересечение фиксировать на непрерывной плоскости н затем отображать его на дискретную плоскость. Непрерывные прямые *AB* и *CD,* соединяющие центры соответствующих пикселов, пересекаются в непрерывной точке, которая может быть отображена в любой из четырех пикселов, отмеченных на рис. 7.13 символом, набранным жирным шрифтом.

Проблема, связанная с этими определениями, не относится к разряду чисто академических. В растровых графических устройствах некоторая прямая, связывающая две точки, обычно воспроизводится при помощи построения к-связного маршрута между ними, и поэтому неединственность может оказаться причиной плохого качества изображения.

Пример 7.1. Воспроизведение линии, связывающей две точки, на растровом графическом устройстве основано на том, что длины отрезков, соответствующих основному направлению, должны быть в максимально возможной степени одинаковыми. На рис. 7.14 пикселы, которые образуют линию, соединяющую точки 0 и 7, снабжены числовыми метками. Очевидно, что точка *f* принадлежит этой линии, однако линия, связывающая точки 0 и *f* и помеченная строчными буквами, не является вопреки определениям евклидовой геометрии подмножеством линии 07. (Пиксел *d* не принадлежит линии 07.) Допустим теперь, что линия 07 является стороной многоугольника, окрашенного в красный цвет, *f* —

точка пересечения с вертикальной линией L и нам необходимо выкрасить все отрезки прямых, расположенные слева от L, в зеленый цвет. В результате получим несколько зеленых пикселов (от a до /), а также один красный пиксел (3) слева от вертикали L.

Наличие сообразного определения для тонких линий представляет также интерес с точки зрения таких задач, как заполнение контура (см. гл. 8) и прореживание области (см. гл. 9). Решение обеих задач предполагает реализацию геометрических операций на дискретной сетке. Ниже мы введем критерий, позволяющий различать тонкие кривые и толстые области. Затем кратко обсудим способы, облегчающие решение геометрических задач.

7.6.1. СЛУЧАЙ МНОЖЕСТВА ПИКСЕЛОВ, НЕ ЯВЛЯЮЩЕГОСЯ КРИВОЙ

Прежде чем перейти к определению тонких областей, определим противоположное понятие.

Определение 7.6. Множество пикселов R называется *полной областью,* если оно содержит более четырех пикселов, его к-контур является простым маршрутом и разность множества и его к-контура является н-связной.

На рис. 7.15, а представлен пример полной области, который, в частности, показывает, что в полную область входят пикселы, расположенные за пределами ее контура. (Это было невозможно, если бы не было предусмотрено условие относительно четырех пикселов.) С топологической точки зрения полная область соответствует открытому множеству.

Грубо говоря, область является полной, если сетка дискретизации достаточно «точна» для того, чтобы ни одна из границ не раздваивалась Условие н-связности является существенным в

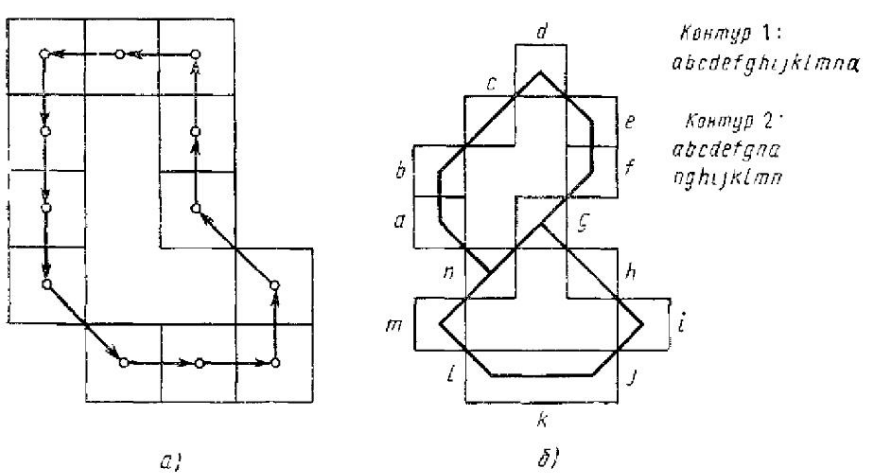
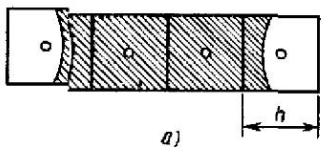


Рис. 7.15. Иллюстрации к определению **7.6:**
а — полная область; *б* — неоднозначность при построении контура, возникающая в результате исключения условия н-связности

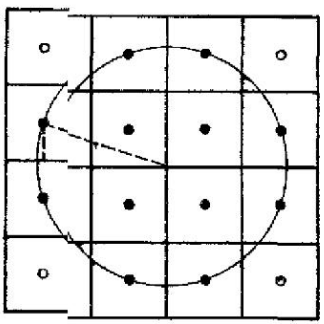
том отношении, что позволяет избежать возникновения неоднозначных ситуаций, типа представленной на рис. 7.15,*б*. Это свойство оказывается полезным в случаях, когда желательно, чтобы форма контура отражала форму объекта. Заполнение таких контуров, кроме того, можно осуществлять с помощью какого-либо простого алгоритма (см разд. 8.3) и, следовательно, указанное свойство имеет существенное значение для машинной графики. Сформулируем достаточное условие, гарантирующее получение исключительно полных областей.

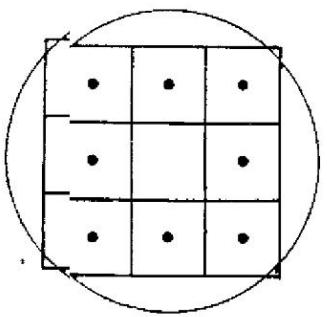
Теорема 7.2. Если диаметр окружности, упоминаемой в определении совместимости, равен sqrt(10)*h*, а не sqrt(2)*h*, то всё множества пикселов, получаемые в результате дискретизации, являются полными

Доказательство. На рис 7.16 представлена наименьшая область, удовлетворяющая условиям данной теоремы, очевидно, что она содержит по меньшей мере, девять пикселов, которые образуют полную область. Области с большими размерами можно представить как объединения таких областей и, следовательно, при их дискретизации также могут быть получены полные множества пикселов. Если R — радиус окружности, то



а)


б)


в)

$$R^2=(h/2)^2 + (3h/2)^2, \text{ или } R = h*sqrt(10)/2.$$

Практическое значение теоремы 7.2 состоит в том, что с ее помощью можно определить размеры элемента сетки дискретизации, и, следовательно, дискретизацию любого заданного набора двухуровневых изображений можно осуществлять таким образом, чтобы в результате получались исключительно полные области. Пусть D — диаметр наименьшей вписанной окружности, удовлетворяющей условиям определения 7.4. В таком случае значение h следует выбирать меньшим или равным D/sqrt(10). С другой стороны, если бы нашей целью было лишь сохранение формы, то значение h можно было бы выбрать меньшим или равным

Рис. 7.16. Иллюстрации к доказательству теоремы 7.2:

а — область, ширина которой меньше 3h, имеет пустую внутреннюю часть, б — область с минимальной шириной sqrt(10)*h* имеет непустую внутреннюю часть, *в* — область с шириной sqrt(10)*h*+ε имеет непустую внутреннюю часть независимо от расположения сетки дискретизации

D/sqrt(2). Отношение двух этих размеров сетки дискретизации равно sqrt(5). Следовательно, для дискретизации, обеспечивающей получение только полных областей, требуется в пять раз больше выборочных точек, чем для дискретизации, при которой не все области оказываются полными. В большинстве прикладных задач пятикратное увеличение необходимого объема памяти и соответственное увеличение времени обработки оказывается слишком дорогим удовольствием. Единственным важным исключением служит машинная графика, в которой высокое разрешение может потребоваться по эстетическим соображениям. Однако даже в машинной графике может возникнуть необходимость работать с областями, которые не являются полными. Мы же используем отсутствие полноты для определения линий и кривых на некоторой дискретной сетке.

7.6.2. СЛУЧАЙ МНОЖЕСТВА ПИКСЕЛОВ, ЯВЛЯЮЩЕГОСЯ КРИВОЙ

Строгое топологическое определение тонких кривых утверждает что они представляют собой непустые множества с пустой внутренней частью, т. е. все точки, из которых они состоят, принадлежат их границе. Итак, введем следующее определение.

Определение 7.7. *Кривой,* или *линейчатой областью* на дискретной сетке называется такое множество пикселов, все элементы которого принадлежат контуру множества.

Это определение допускает существование множеств, имеющих толщину 2; соответствующий пример приведен на рис. 7.17. Другое определение тонких множеств предусматривает существование единственного маршрута для любой пары точек такого множества. При таком подходе пример, приведенный на рис. 7.17, не будет отнесен к разряду тонких множеств. Установлено, что при решении большинства прикладных задач предпочтительно пользоваться определением 7.7, а не альтернативным ему. Более подробно эта проблема будет рассмотрена в гл. 9.

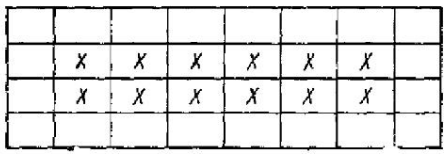


Рис. 7.17. Пример тонкого множества в смысле определения 7.7

7.7. КРАТНЫЕ ПИКСЕЛЫ

Хотя мы теперь и располагаем неким определением тонкого множества, мы еще не покончили с этой проблемой. В самом деле, мы можем столкнуться со смешанными областями, допускающими представление в виде объединения полных и линейчатых областей.

В таком случае необходимо иметь определение, применимое к подмножествам области, а также (путем его расширения) к отдельным пикселам.

Мы уже убедились в том, что пикселы, значение метки которых после обхода контура оказывается больше двух, играют особую роль. Рассмотрим их свойства после ввода нескольких определений.

Определение 7.8. *Ко-соседями* пиксела, принадлежащего контуру C, называются пикселы, являющиеся для этого пиксела предыдущим и следующим элементами при обходе, осуществляемом в соответствии с алгоритмом 7.1.

Заметим, что два ко-соседа пиксела не обязательно должны быть разными пикселами, но они всегда различны в случаях, когда контур представляет собой простой маршрут. Пример, иллюстрирующий введенное определение, дан на рис. 7.18.

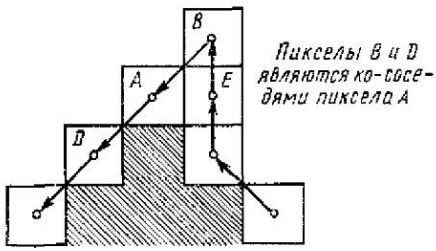


Рис. 7.18. К определению ко-соседей для некоторого пиксела контура; пикселы B и D являются ко-соседями пиксела A, а пиксел E таковым не является; пикселы E и A являются ко-соседями пиксела B

Рис. 7.19. Содержательная иллюстрация понятия «кратный пиксел»:

A соответствует развороту границы [условие (б)]; B — двум непересекающимся дугам, отображенным на один и тот же пиксел [условие (а)], а C и D являются соседними пикселами, на которые отображены непересекающиеся дуги границы [условие (в)]

Определение 7.9 Пиксел называется *кратным* при выполнении одного или нескольких из следующих условий:

(а) данный пиксел просматривается более одного раза в процессе построения контура;

(б) данный пиксел не имеет соседей во внутренней части соответствующей области;

(в) данный пиксел имеет, по меньшей мере, одного н-соседа, который принадлежит контуру, но не является его ко-соседом.

Смысл этого определения иллюстрируется на рис. 7.19. Кратными оказываются те пикселы, на которые попадают две дуги контура, а также пикселы, на которых происходит «разворот» контура. Первые два условия легко проверяются с помощью разметки, предусмотренной в процедуре *TRACER* (см. алгоритм 7.1). В самом деле, по окончании построения контура «обычные» пикселы

контура будут снабжены меткой, значение которой будет равно 2, а кратные пикселы, удовлетворяющие условию (а), будут иметь метки, значения которых будут больше 2. Условия (б) (отсутствие соседей со значением метки 1) и (в) (наличие н-соседа со значением метки, равным или больше 2, при условии, что он не является ко-соседом) могут быть проверены с помощью второго обхода. Несмотря на существенную простоту обнаружения кратных пикселов при помощи обходов контура, этот метод может оказаться неприемлемым при решении тех прикладных задач, в которых целесообразно использовать параллельную обработку. Поскольку проверка условия (в) существенным образом зависит от этой последовательной процедуры обхода, то определение 7.9 начинает создавать проблемы. Интуитивно кажется, что последовательность не должна быть столь уж важной, так как нас интересуют пикселы, на которые приходится две или несколько дуг контура, или те пикселы, на которых дуги изгибаются, т. е. конфигурации, не зависящие от порядка обхода.

Продолжим изучение пикселов, не зависящих от порядка обхода. Если в процессе построения контура некоторого множества R некоторый пиксел просматривается более одного раза, то это может происходить только из-за отсутствия способа прохождения из одной части множества $R(R_1)$ в другую его часть (R_2), минуя данный пиксел. Следовательно, при удалении данного пиксела число связности множества R уменьшится, по крайней мере, на единицу. И наоборот, любой пиксел, удаление которого из множества R уменьшает его число связности, должен в процессе построения контура этого множества просматриваться более одного раза. Установить, существен ли некоторый пиксел с точки зрения связности соответствующей области, можно, изучив восемь его соседей. Нетрудно показать, что пиксел P является существенным для связности соответствующей области в том и только том случае, если конфигурация его окрестности соответствует образцам, приведенным на рис. 7.20 (в том числе и полученным в результате их поворота на 90° в любом направлении). Таким образом, некоторый пиксел можно считать кратным в том и только том случае, если его окрестность имеет конфигурацию, соответствующую одному из представленных образов. На этом (и последующих) рисунке значения пикселов записываются следующим образом.

Определение 7.10. В качестве меток, присваиваемых некоторому пикселу в процессе обхода контура, используются сле-

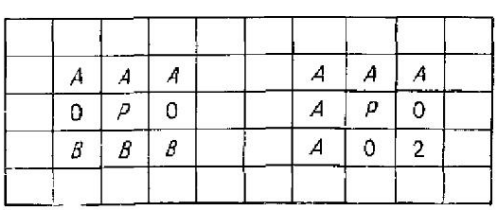


Рис. 7.20. Конфигурация окрестности пиксела, являющегося существенным для связности соответствующей области

дующие числа: 1— для пикселов, находящихся внутри области; 2 — для пикселов контура, осматриваемых однократно; 3 или большее число — для пикселов контура, осматриваемых более одного раза. С помощью числа, за которым следует знак (+), обозначается пиксел, метка которого, по крайней мере, не меньше этого числа. Для обозначения пикселов, не принадлежащих соответствующему множеству, используется 0. Буквы, за исключением *X*, обозначают, что данный пиксел может иметь любую ненулевую метку. Метка *X* заменяет любое число (условие не имеет значения). Группа пикселов, помеченных одним и тем же символом (например, *A*), характеризуется тем свойством, что по меньшей мере один из этих пикселов имеет метку, значение которой больше нуля. Звездочка (*) используется для обозначения пиксела, метка которого может иметь любое отличное от 1 значение.

Таким образом, в эталонных конфигурациях, приведенных на рис. 7.20, по меньшей мере один из трех пикселов, обозначенных символом *A*, имеет ненулевую метку (левый эталон) и по меньшей мере один из трех пикселов, обозначенных символом *B*, имеет ненулевую метку. Отметим, что принадлежность пиксела контуру можно установить, не прибегая к обходу. Достаточно установить, что данный пиксел имеет н-соседа, снабженного нулевой меткой.

Итак, нами доказано следующее.

Утверждение 7.2. Условие (а) определения 7.9 можно заменить требованием, чтобы восьмиэлементная окрестность пиксела соответствовала хотя бы одной из эталонных конфигураций, приведенных на рис. 7.20 (или полученных в результате их поворота на 90° в любом направлении).

Условие (б) легко поддается проверке с помощью какого-либо параллельного алгоритма. Ни один из восьми пикселов-соседей не может иметь метку, равную 1. Это обеспечивает сохранение одиночных (т. е. пикселов, не имеющих ни одного соседа) и концевых (пикселов, имеющих ровно одного соседа) точек. Таким образом, проверка соответствуя второму эталону на рис. 7.20 может быть упрощена, если допустить, чтобы все обозначенные символом *A* пикселы имели нулевые метки. Заметим, что условие (б) обеспечивает также сохранение всех линий, имеющих ширину, равную двум (типа линии, изображенной на рис. 7.17).

Нам осталось рассмотреть условие (в). Если пиксел удовлетворяет этому условию, а также условию (а) или (*б*), то в любом случае он будет отнесен к разряду кратных пикселов. Следовательно, можно попытаться найти критерий, более жесткий, чем условие (в), однако обладающий той особенностью, что отсеянные по нему пикселы «отлавливаются» при помощи двух других условий. Покажем, что для проверки таких пикселов вместо условия (в) можно использовать эталонную конфигурацию пикселов размерами 3x3. Для этого в качестве исходной используется эталонная конфигурация пикселов 4X4 (т. е. больших размеров) и рассматриваются все допустимые в ней конфигурации, порождаю-

щие условие (в). Общий случай такой конфигурации представлен на рис. 7.21.

Без потери общности можно считать, что c_2 - осматриваемый в данный момент (текущий) пиксел, снабженный меткой 2, а c_1 — его н-сосед с меткой 0. (Напомним, что все пикселы, входящие в контур, имеют такого соседа.) Другие конфигурации можно получать с помощью поворота исходной конфигурации на 90°. Поскольку пиксел c_2 удовлетворяет условию (в) определения *7.9,* он также должен иметь н-соседа с меткой, не меньшей 2.

Утверждение 7.3. Любой пиксел, удовлетворяющий условию (в) определения 7.9 и не удовлетворяющий условию (а) или (б), входит в конфигурацию, обозначаемую кодом 022 + и ориентированную по вертикали или горизонтали.

	a_0		
a_1	a_2	a_3	a_4
c_1	c_2	c_3	c_4
b_1	b_2	b_3	b_4

Рис. 7.21. Разметка пикселов, использованная при введении альтернативной формы определения 7 9

Доказательство. Докажем это свойство для пиксела c_3. Начнем с рассмотрения пиксела C_3 и покажем, что, если его метка не равна 2+ (т. е. данное утверждение не выполняется для горизонтальной конфигурации), то пиксел c_2 входит в некоторую вертикальную конфигурацию, обладающую указанным свойством. Если метка пиксела c_3 имеет нулевое значение, то пиксел c_2 либо многократно осматривается, либо является концевой точкой и, следовательно, в соответствии с условием (а) или (б) относится к категории кратных пикселов. Пусть его метка равна 1. Это означает, что пикселы a_3, c_4 и b_3 не могут иметь нулевые метки, поскольку в противном случае пиксел c_3 должен входить в контур. Так как мы приняли, что условие (в) выполняется, значение метки пиксела a_2 или b_2 должно быть не менее 2. Учитывая симметричность, необходимо рассмотреть лишь один из этих пикселов. Пусть пиксел a_2 имеет значение 2 +. Если метка пиксела a_1 равна 0, то пиксел a_2 будет ко-соседом пиксела c_2, из чего следует, что метка пиксела b_2 должна также иметь значение не меньше 2, а пиксел b_1 должен иметь ненулевую метку (в силу симметричности конфигурации). Итак, продолжим доказательство, считая, что пиксел a_1 имеет ненулевую метку. В данном случае пиксел a_2 входит в контур; его н-соседями являются пикселы c_2, a_1 и a_3, имеющие ненулевые метки. Следовательно, метка пиксела a_0 должна быть нулевой, и, таким образом, пикселы a_0, a_2 и c_2 образуют конфигурацию, имеющую код 022 +•

Этот результат позволяет без потери общности допустить, что пиксел c_3 (см. рис. 7.21) имеет метку, значение которой не меньше 2. Теперь перейдем к рассмотрению допустимых конфигураций пикселов a_1, a_2 и a_3 и b_1, b_2 и b_3.

Случай 1. Пикселы a_3 и b_3 имеют ненулевые метки. При этом метка пиксела c_4 должна быть нулевой, так как в противном случае пиксел c_3 не мог бы входить в контур. Если бы все пикселы

a_1, a_2, b_1 и b_2 имели нулевые метки, то метки пикселов a_3 и b_3 должны были бы быть равны 2+ и пиксел c_2 должен был бы в соответствии с условием (б) (отсутствие соседей с единичной меткой) быть отнесен к разряду кратных. В таком случае должна возникнуть конфигурация, приведенная на рис. 7.22,*а*, причем по меньшей мере один из четырех пикселов должен иметь ненулевую метку. Число допустимых конфигураций с таким кодом очень невелико и можно убедиться в том, что для всех них условие (в) выполняется. (Пример одной из таких конфигураций приведен на рис. 7.22,*б*.)

A	A	P	X			0	0	2	X
0	2	2+	0			0	2	2	0
A	A	Q	X			2	1	1	2

а) *б)*

Рис. 7.22

*	*	0	X
0	2	2+	X
*	*	0	X

Рис. 7.23

Рис. 7.22. Иллюстрация к случаю 1:
а — общий и б — частный случай

Рис. 7.23. Иллюстрация к случаю 2: ни один из пикселов с метками 2 и 2+ не является существенным для связности, однако при удалении их обоих число связности изменяется

Случай 2. Пикселы a_3 и b_3 имеют нулевые метки. В таком случае нетрудно убедиться в том, что пиксел c_2 не может иметь соседей с единичной меткой и, следовательно, должен классифицироваться в соответствии с условием (б). На рис. 7.23 приведена конфигурация (и ее код), соответствующая этому случаю.

Случай 3. Пиксел a_3 имеет ненулевую метку, а пиксел b_3 — нулевую. (Случай, когда метка пиксела a_3 — нулевая, а значение метки пиксела b_3 больше нуля, аналогичен и мы не будем его обсуждать). Если пикселы b_1 и b_2 имеют нулевые метки, то контур ведет от пиксела c_2 к пикселу c_3 и пиксел c_2 не может быть кратным, если он не удовлетворяет условию (в). Соответствующая конфигурация и ее код приведены на рис. 7.24,*а* а на рис. 7.24,*б* представлен частный случай. Если метки пикселов b_1 и b_2 не равны нулю одновременно, а оба пиксела a_1 и a_2 имеют нулевые метки, то пиксел c_2 не имеет соседей внутри области и должен классифицироваться в соответствии с условием (б).

Мы исчерпали все допустимые конфигурации, возникающие в прямоугольнике пикселов размерами 3X4. Если условие (в) вообще выполняется, то оно должно быть выполнено на такой области. Итак, нами доказано следующее утверждение.

Утверждение 7.4. Проверка условия (в) эквивалентна поиску конфигурации вида, приведенного на рис. 7.25 (а также конфигу-

раций, полученных в результате поворота указанной конфигурации на 90° в произвольном направлении). По меньшей мере, один из обозначенных символом C пикселов должен иметь ненулевую метку; если ненулевую метку имеют оба пиксела, обозначенные символом C, то метки пикселов, обозначенных символами A и B, могут быть любыми, в противном случае по меньшей мере один из каждой пары пикселов A и B должен иметь ненулевую метку.

X	X	P	X			2	2	1	X
0	2	2+	X			0	2	2	1
0	0	0	X			0	0	0	2

а) *б)*

Рис. 7.24

A	A	C
0	2	2+
B	B	C

Рис. 7.25

Рис. 7.24. Иллюстрация к случаю 3:
а — общий и б — частный случаи

Рис. 7.25. Конфигурация, наличие которой эквивалентно выполнению условия (в)

На рис. 7.26 приведено несколько характерных конфигураций пикселов (вместе с соответствующими кодами). Если осматриваемый пиксел является кратным, то он обозначен символом, набранным жирным шрифтом.

0	2	1	1	1	1			0	2	1	1	1	1
0	0	2	1	1	1			0	0	2	1	1	1
0	0	0	**P**	2	1			0	0	0	**P**	2	1
0	0	0	2	0	2			0	0	0	0	0	2
0	0	0	0	2	1			0	0	0	0	2	1

Рис. 7.26. Пример кратного пиксела *(P)*, удовлетворяющего условию (в), причем этот пиксел не удовлетворяет ни одному из двух других условий *(слева)*; пример, когда для пиксела P условие (в) не выполняется *(справа)*

Итак, нами получено другое определение для кратных пикселов.

Теорема 7.3. Пиксел является кратным, если он удовлетворяет хотя бы одному из следующих условий:

(а) его окрестность соответствует одной из конфигураций, представленных на рис. 7.20 (или конфигурациям, полученным их поворотом на 90° в произвольном направлении); пикселы второй конфигурации, обозначенные символом A, могут иметь метки с произвольными значениями;

(б) пиксел имеет хотя бы одного соседа с ненулевой меткой или не имеет соседей с метками, равными единице;

(в) окрестность пиксела соответствует конфигурации, представленной на рис. 7.25 (или конфигурациям, полученным ее поворотом на 90° в произвольном направлении).

Очевидно, что перечисленные условия можно проверять как в последовательном, так и параллельном режимах. Сформулируем результат, связывающий понятие кратных пикселов с тонкими линиями и кривыми.

Утверждение 7.5. Любая линейчатая область состоит исключительно из кратных пикселов. И, наоборот, если некоторая область содержит только кратные пикселы, то она является линейчатой.

Доказательство. Поскольку данное множество не содержит никаких пикселов, за исключением составляющих его контур, ни один из пикселов контура не имеет соседей внутри области, и, следовательно, условие (б) определения 7.9 справедливо. Доказательство обратного утверждения тривиально, поскольку любой кратный пиксел всегда входит в контур.

Проведенный нами анализ имеет значение для обнаружения линий и выполнения операций над ними на дискретной сетке. Мы не касались задачи построения соответствующих отображений, исходя из уравнений прямых и кривых. Эта тема будет затронута в гл. 10

7.8. ВВЕДЕНИЕ В ЗАДАЧИ АНАЛИЗА ФОРМЫ ОБЪЕКТОВ

Анализ формы объектов представляет собой одну из основных задач распознавания образов и имеет определенное значение для решения задач машинной графики в интерактивном режиме. Анализ формы оказывается полезным во всех случаях, когда требуется принять некоторое решение на основе формы наблюдаемых объектов. Точное психофизическое определение того, что такое форма, выходит за пределы задач данной книги. Естественно, термин «удлиненный» или «острый угол» характеризуют форму. Воспользовавшись определением через отсутствие некоторых признаков, можно указать, что форма представляет собой информацию, содержащуюся в двухуровневом изображении и не учитывающую цвет соответствующих областей. Хотя область действия этого определения ограничивается силуэтами, оно оказывается адекватным при решении многих прикладных задач, в частности, при распознавании буквенно-цифровых символов.

Можно выделить два подхода к распознаванию формы объектов. При использовании первого человек рассматривает объект в целом и принимает решение, исходя из его общей структуры. Обычно именно таким образом проводится распознавание стилизованных рукописных символов, особенно китайских, когда идентифицируются штрихи и другие элементарные блоки, из которых строится соответствующий символ. При другом подходе исследуется контур силуэта: обычно отыскиваются углы, выступы, впадины и

другие точки с высокими значениями кривизны. Примерами использования такого подхода является распознавание силуэтов профилей человеческих лиц, а также обнаружения дефектов очертаний компонентов схем, выполненных на печатных платах. [1]. Естественно, во многих случаях приходится использовать оба подхода — в технических чертежах, например, встречаются линии и символы, поддающиеся распознаванию на основе анализа их структуры, а также окружности и шестиугольники, для различения которых необходимо учитывать их контуры. В прошлом большинство методологий ориентировалось на использование одного из двух этих способов и поскольку теоретически допустимо их применение к любому объекту, они использовались и в тех ситуациях, для работы с которыми они не очень подходили. Идеально было бы иметь комбинированный метод, который мог бы автоматически настраиваться на режим, наиболее подходящий для работы с изучаемым объектом.

Здесь мы ограничимся лишь обсуждением применения простых методов анализа контура для описания объектов, выделяемых на изображениях класса 2. Структурные методы подвергаются краткому обсуждению в гл. 9. В анализе формы кривизна является важным признаком, причем не только при непосредственном, но и при опосредованном ее использовании. К сожалению, прямое измерение значения кривизны не всегда возможно из-за наличия шума. Соответствующая формула, приводимая в большинстве учебников по математическому анализу, предусматривает взятие второй производной и потому ею невозможно пользоваться при решении практических задач. С другой стороны, можно оценить значение радиуса кривизны с помощью геометрической конструкции, приведенной на рис. 7.27. Действительно, если A, B и C — точки, расположенные на некоторой кривой, M и N — точки, расположенные посередине отрезков AB и BC соответственно, и K — точка, в которой пересекаются нормали к точкам M и N, то длина отрезка BK, R, равна радиусу некоторой окружности, проходящей через точки A, B и C. Если φ — угол ABC, ψ обозначает его дополнение и длины отрезков AB и BC равны $2e$, то можно непосредственно получить, что

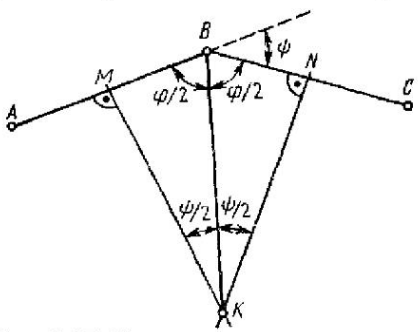


Рис. 7.27. Конструкция, использованная для оценки радиуса кривизны

$$R = \frac{e}{\cos(\varphi/2)} = \frac{e}{\sin(\psi/2)} \qquad (7.1)$$

[1] Этот подход не следует смешивать с так называемым *гештальтистским* восприятием формы, посредством которого, согласно представлениям психологов, распознавание формы некоторой фигуры производится без обращения к анализу ее частей.

(см. задачу 7.7). Из чертежа, приведенного на рис. 7.27, и формулы (7.1) очевидно, что R является растущей функцией угла φ. Если угол ψ); мал, что бывает часто, то кривизна c, определяемая как величина, обратная радиусу кривизны, будет приближенно определяться следующим образом:

$$c = \psi/2\, e = (\pi - \varphi)/2\, e. \qquad (7.2)$$

Для того чтобы ограничить влияние шума, можно вычислять значение кривизны не по трем последовательным точкам, а по таким трем точкам, для которых расстояние e достаточно велико относительно пространственной частоты шума. Известно множество вариантов этого метода, однако мы не будем здесь на них останавливаться. Для косвенного определения максимумов кривизны можно воспользоваться аппроксимацией многоугольниками, так как их углы обычно располагаются около этих максимумов.

Дальнейший анализ контура может проводиться несколькими способами. Простейшая методология предусматривает получение несложного представления контура, например, в цепном коде. При использовании более развитой методологии контур аппроксимируется участками гладких кривых (например, B-сплайнами). Последнее предпочтительно в случаях, когда данные зашумлены, а также при использовании признаков, отражающих особенности значительной части контура. Первый подход более уместен при работе с данными, отличающимися низким уровнем шума, и использовании локальных признаков. Широкое применение аппроксимации многоугольниками объясняется не только связанной с ней возможностью обнаруживать максимумы кривизны, но и тем, что ее реализация оказывается проще реализации других методов построения кривых по точкам. Обсуждение этих методов мы отложим до гл. 11 и 12, в которых рассматриваются сплайны и методы аппроксимации.

Рассмотрим выделение локальных признаков с помощью дифференциального цепного кода (см. подразд. 1.2.3), поскольку он позволяет строить описания, не зависящие от ориентации. Если контур достаточно гладкий, то описание будет состоять только из символов 0 и ±1. Присутствие в нем символов ±2 означает наличие угла, равного 90°, а символов ±3 — угла, равного 45°. Несложно связать появление этих символов с максимумами кривизны, однако для определения положения всех максимумов этого недостаточно. Например, последовательность 01110 также представляет угол, равный 45°.

Интерес представляют дуги четырех классов: прямые, углы, кривые, аппроксимирующие дуги окружности, и пазы. Прямая, направление которой совпадает с одним из направлений цепного кода, имеет вид 0^n. Если ее направление не совпадает ни с одним из направлений цепного кода, то она имеет вид $(O^m{+}1{-}1)^k$ или

$(O^m — 1 + 1)^{k\ 1}$. Наличие шума может привести к тому, что кодовая последовательность, представляющая реальную прямую, окажется не столь регулярной, однако в ней всегда будет присутствовать пара «+1 - 1» или « -1 +1». Представление дуги окружности с другой стороны, будет иметь вид $(O^m+1)^k$ или $(O^m—1)^k$, причем основной особенностью этого представления является наличие одиночных символов ±1, знак которых для конкретной кривой фиксирован. Представление угла будет включать либо одно из больших значений кодовых символов, либо последовательность единиц с одинаковыми знаками. Зубец представляет собой заостренную впадину или заостренный выступ. Зубцы имеют существенное значение при решении некоторых прикладных задач, в частности, в тех случаях, когда контуры объектов должны быть очень гладкими и, следовательно, наличие зубцов означает появление дефекта. Представление зубца в цепном коде образует последовательность больших значений кодовых символов, присоединенную к 0 (или самое большее к ±1). На рис. 7.28 приведены примеры зубцов.

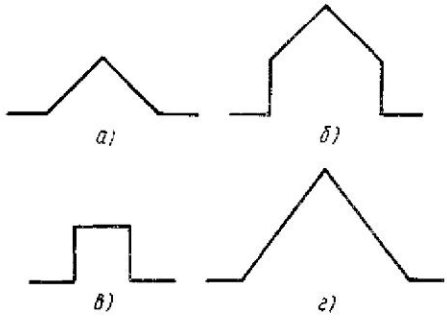


Рис. 7.28. Примеры зубцов. Соответствующие представления зубцов, записанные в дифференциальном цепном коде, имеют вид: *а* — ... 0 + 1 —2 +1 0...; *б* — ...0 + 2 — 1—2 —1+2 0...; *в* — ...0 + 2 — 2—2 + 2 0..., *г* — ...0+ 10 — 20+ 1 0...

Для выделения таких признаков можно предложить простые алгоритмы, причем их применимость определяется тем, сколь мало зашумлены исходные данные. Удачным, в частности, является метод, предусматривающий задание регулярных выражений или (что эквивалентно) конечных автоматов, соответствующих подобным признакам. Например, следующее выражение обеспечивает распознавание части зубцов, изображенных на рис. 7.28:

О (+1 или +2) (цепочка, образованная максимум двумя отрицательными символами) (+1 или +2) 0.

Более сложную задачу составляет распознавание того, представляет ли некоторая длинная последовательность некоторую прямую или некоторую дугу окружности. Для обнаружения подобных крупномасштабных признаков лучше пользоваться иными методами, например типа построения кривых по точкам (см. главу 12).

[1] На самом деле, значение *m* может при переходе от одного отрезка к другому изменяться на единицу. Детальное описание представления прямых с помощью цепных кодов можно найти в докладе [7.13].

7.9. БИБЛИОГРАФИЧЕСКИЕ ЗАМЕЧАНИЯ

Задачи, связанные с определением связности и других топологических понятий на дискретной сетке, впервые были систематизированы в работах [7.6, 7.10]. Построение контура представляет собой достаточно простую процедуру, если не считать проблем, связанных с дискретной связностью. Обсуждение этой темы можно найти в статье [7.11]. В статье [7.5] описывается алгоритм, не предусматривающий разметку изображения. В статье [7.7] представлен алгоритм, выполняющий построение контура на основе кодирования изображения с переменной длиной кодовой последовательности.

Подробное изложение методов анализа формы объектов выходит за пределы задач нашей книги. При необходимости читатель может обратиться к литературе по распознаванию образов, в которой этот предмет обсуждается детально; книги [2.3, 3.7, 7.2, 7.3] представляют собой научные монографии, в работах [7.8, 7.9] даны обзоры литературы. Читатели, интересующиеся применением изложенных методов в конкретных прикладных областях, должны обратиться к литературе, посвященной соответствующим приложениям. Например, обзор первых работ по распознаванию символов имеется в книге [7.12]. Во многих работах можно найти примеры задач, связанных с анализом контуров. Роли полостей при описании рукописных символов уделено внимание в целом ряде работ и, в частности, в статье [7.1] и докладе [7.14].

7.10. ЗАДАЧИ

7.1. Найдите фурье-образ прямоугольного сигнала, имеющего минимальную ширину полуволны H, т. е. каждый интервал, в котором значение сигнала пос тоянно, насчитывает, по меньшей мере, H единиц длины. Решите эту задачу также для случая, когда изменения значения сигнала происходят в точках, разделенных промежутками, длина которых равна целым кратным некоторой величины h. Сравните два полученных результата преобразования и исследуйте, каким образом значение отношения H/h влияет на их близость.

7.2. Постарайтесь оценить значение интервала выборки, необходимого для дискретизации текста, помещенного на этой странице.

7.3. Постройте алгоритм для подсчета числа цветных пятнышек (связных областей некоторого заданного цвета), имеющихся на изображении класса 2.

7.4. При решении практических задач часто возникает необходимость уве личения или уменьшения некоторого двухуровневого изображения. Например, может потребоваться воспроизвести в увеличенном или уменьшенном вдвое масштабе набор двухуровневых матриц, представляющих буквенно-цифровые символы. Можно ли построить такой алгоритм, который, выполняя предписан ные масштабные преобразования, обеспечивает сохранение формы символов? В частности, определите для заданного множества наибольший допустимый коэффициент уменьшения масштаба.

7.5. Реализуйте алгоритм 7.2 и используйте его для подсчета числа цвет ных пятнышек, представляющих собой полные, линейчатые и комбинированные множества.

7.6. Пусть R — связное множество пикселов и $C(R)$—пикселы, образую щие его контур. Покажите, что при удалении всех пикселов, принадлежащих $C(R)$, за исключением кратных, оставшиеся пикселы образуют множество.

являющееся не только связным, но и топологически эквивалентным множеству R. Что Вы могли бы сказать о подобии форм этих областей?

7.7. Докажите справедливость формулы (7.1).

7.8. Введите некоторую меру гладкости контура, основанную на использовании дифференциального цепного кода (например, доля ненулевых значений в их общем числе). После этого получите такое решение задачи 7.4, которое обеспечивает сохранение постоянного или практически постоянного значения этой меры.

Глава 8 ЗАПОЛНЕНИЕ КОНТУРА

8.1. ВВЕДЕНИЕ

Определение внутренней части области, контур которой задан, является одной из самых распространенных задач машинной графики и анализа изображений. В сущности, речь идет о преобразовании изображения класса 3 в изображение класса 2. Например, любой алгоритм заштриховки решает эту задачу. Многие алгоритмы, применяемые в распознавании образов, предусматривают вычисление интеграла по площади некоторой области, для чего они должны быть снабжены информацией о внутренней части области. При использовании фотонабора комплект шрифта часто задается контурами, которые затем заполняются для получения искомого отпечатка. Задачу заполнения можно решать множеством способов, разделив их на два обширных класса. Первый из них предполагает наличие точного описания контура как прямоугольника, и решение о том, какие части плоскости лежат внутри рассматриваемой области, принимается, по существу, на основе анализа уравнений, задающих соответствующие линии. Эти методы, изложенные в разд. 8.2, можно было бы определить как методы многоугольника, однако мы будем пользоваться термином «заполнение области, ограниченной сторонами многоугольника». Методы второго класса предусматривают отображение заполняемого контура на дискретную плоскость и определение положения внутренней части области на основе значений яркости пикселов. Методы заполнения контура, основанные на анализе значений пикселов, обсуждаются в разд. 8.3—8.5.

Кроме того, методы заполнения можно различать, исходя из того, какой принцип применяется в них для установления принадлежности точки внутренней части многоугольника. В некоторых алгоритмах используется проверка на четность, в других — критерий связности.

Алгоритмы заполнения области, использующие проверку на четность, основываются на том, что произвольная прямая пересекает любую замкнутую кривую (типа контура некоторой области) четное число раз (рис. 8.1). Если известно, что первая

точка соответствующей линии лежит вне рассматриваемой области, то, выполнив обход этой линии, можно установить путем подсчета числа пересечений, какие ее отрезки расположены внутри области. Если число пересечений — нечетное, то соответствующий отрезок расположен внутри области (отрезки *AB* и *CD* на рис. 8.1), в противном случае — вне области (отрезок *BC* на рис. 8.1). Все алгоритмы заполнения области между сторонами многоугольника предусматривают проверку на четность. Этот принцип можно использовать и в алгоритмах заполнения области на основе анализа значений пикселов, однако при этом возникают серьезные проблемы. Может оказаться, что точки, принадлежащие двум или нескольким сторонам, отображены на один и тот же пиксел (см. разд. 7.6), что приводит к ошибкам в подсчете числа пересечений. Следует также соблюдать осторожность при работе с линиями, являющимися касательными к контуру: при подсчете числа пересечении точку касания следует учитывать дважды. Нами предложены способы, по крайней мере частичного, преодоления этих проблем (см. разд. 8.3).

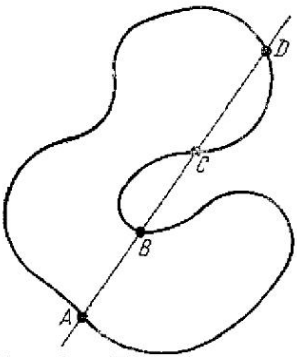


Рис. 8.1. Пример использования принципа проверки на четность для определения точек, лежащих внутри замкнутой кривой

Заполнение области по критерию связности предполагает, что помимо контура задана некоторая точка, расположенная внутри области (затравка). В таком случае совершается обход плоскости, целью которого является обнаружение всех пикселов, достижимых из затравочной точки без пересечения контура. Моделью этой процедуры может служить обход графа, предусматривающий использование какой-либо из рассмотренных в гл. 6 структуры данных. Заполнение области по критерию связности уместно осуществлять лишь с помощью алгоритмов, предусматривающих анализ значений яркости пикселов, поскольку этот критерий предполагает произвольный доступ к пикселам, находящимся внутри области. Операцию заполнения лучше всего проводить в памяти обновления изображения растровых графических устройств или использовать для ее осуществления точную копию памяти обновления изображения, хранящуюся в оперативной памяти. Основным достоинством заполнения области по критерию связности является устойчивость этой процедуры к нарушениям регулярности контура, сохраняющаяся в случаях, когда контур представляет собой замкнутую кривую. Основным недостатком этого метода заполнения является необходимость знать заранее точку, расположенную внутри соответствующей области. Таким образом, процедура хорошо приспособлена для реализации на интерактивных графических устройствах: пользователь может нарисовать контур (по возможности, вполне регулярный), указать внутри него точку (соз-

дав таким образом затравочную точку) и задать цель — заполнить контур. В разд. 8.4 нами будут рассмотрены алгоритмы, реализующие критерий связности.

Используя различные комбинации критериев связности и проверки на четность, можно получать новые алгоритмы. Подобные комбинированные критерии легко реализуются с применением какой-либо общепринятой структуры данных. Оказывается, что граф смежности строк контура (К-ГСС) подходит для реализации обеих критериев. Очевидно, что эта структура данных годится для реализации алгоритма заполнения области, основанного на анализе значений яркости пикселов и использующего критерий четности, поскольку главным объектом этого алгоритма является контур. При реализации алгоритмов, в которых используется критерий связности, в качестве структуры данных обычно выбирается граф смежности строк внутренней части области (В-ГСС), однако утверждение 6.5 указывает способ, позволяющий применять при реализации этих алгоритмов и К-ГСС. Если допустить, что внутренняя часть области характеризуется н-связностью, контур— к-связностью, внутренняя часть области окрашена в цвет X и пикселы контура — в цвет Y, то утверждение 6.3 можно переформулировать следующим образом.

Утверждение 8.1. Если вершина В-ГСС имеет порядки (m, n), $n>1$, то в К-ГСС на строке, смежной снизу со строкой рассматриваемой вершины В-ГСС, найдется вершина, имеющая порядки $(0, d)$. Аналогичным образом, при $m>1$ в К-ГСС на строке, смежной сверху со строкой рассматриваемой вершины В-ГСС, найдется вершина, имеющая порядки $(d, 0)$. Справедливо также обратное утверждение.

Следовательно, вместо проверки порядков вершин В-ГСС (что требуется для обхода графа) можно проверять порядки вершин К-ГСС. Выбор такого способа обладает двумя практическими преимуществами. Во-первых, появляется возможность комбинировать алгоритмы, основанные на критериях четности и связности, а во-вторых — возможность увеличения скорости решения. Определение порядка вершин графа требует одновременного просмотра трех строк. В случае В-ГСС эти строки могут быть достаточно длинными, в то время как интервалы контура обычно оказываются существенно более короткими. В связи с возможностью комбинации алгоритмов мы будем пользоваться К-ГСС при применении алгоритмов обоих типов.

8.2. ЗАПОЛНЕНИЕ ОБЛАСТИ МЕЖДУ СТОРОНАМИ МНОГОУГОЛЬНИКА

Заполнение области между сторонами многоугольника (методом многоугольника) называют также *заполнением области на основе преобразования развертки,* хотя иногда эти термины употребляются и в другом смысле. Стороны многоугольника (границы) упорядочиваются и помечаются таким образом, чтобы обес-

печивалось сохранение всей топологической информации. Затем с помощью простого однопроходового алгоритма проводится проверка на четность и собственно процедура заполнения не требует произвольного доступа к пикселам. Ее с равным успехом можно выполнить как на растровых, так и на векторных графических устройствах. Особенно удобно ее использовать при решении таких прикладных задач, в которых один и тот же контур воспроизводится многократно (например, при фотонаборе), поскольку предварительная сортировка и разметка, требующие значительных затрат вычислительных ресурсов, должны выполняться лишь один раз.

Исходными данными для нас служит последовательность сторон (границ) многоугольника, аппроксимирующего контур. Для каждой стороны заданы координаты x_i и y_i угла, которому соответствует максимальное значение y или минимальное значение x, если эта сторона расположена горизонтально, а также значения разностей • x и • y, позволяющие определить координаты второго угла при помощи суммирования этих разностей с координатами первого угла. Алгоритм заполнения области между сторонами многоугольника в первую очередь проводит сортировку сторон в соответствии со значениями координаты y_i, а в случае равенства значений y_i — в соответствии со значениями координаты x_i. Если равенство обнаруживается и в этом случае, то используется значение • x и затем • y. При использовании для сортировки значений этих разностей прежде всего выбирается сторона с наименьшим значением[1]. Так, в примере, приведенном на рис. 8.2, сторона AB будет взята раньше, чем сторона AC, поскольку разность • x; отрицательна для AB и положительна для AC. После этого алгоритм отыскивает максимальные значения по координате y и находит точки пересечения сторон многоугольника с прямыми, параллельными оси x. В примере на рис. 8.2 прямая, проходящая

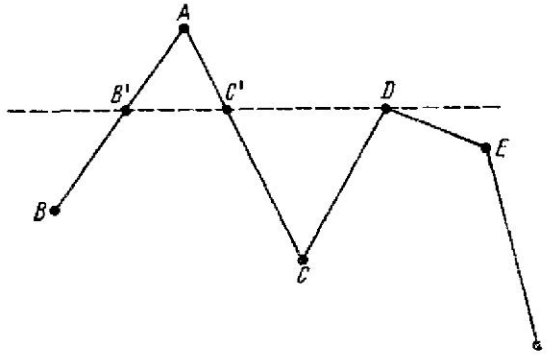


Рис. 8.2. Заполнение области внутри границ: каждый горизонтальный слой содержит четное число границ и внутренняя часть области заключена между первой и второй, третьей и четвертой и т. д. границами

[1] Наименьшее в алгебраическом смысле, т. е. любая отрицательная величина считается меньше любой положительной.

через точку D, образует два таких пересечения — в точках B' и C'. Изображение разделяется этими прямыми на слои, каждый из которых содержит четное число границ, причем внутренняя часть области заключена между границами с нечетным и четным номерами.

Границы, прошедшие сортировку, заносятся в список очередности; для заполнения области используется текущий список, содержащий границы, находящиеся в каждом слое. Начальным значением y в этой процедуре является максимальное и она продолжается вплоть до достижения минимального значения y. В списке очередности предусмотрены метки, позволяющие задавать следующие условия.

а. Число границ, подлежащих передаче в текущий список: одна, если рассматриваемая точка находится на стороне многоугольника; две, если рассматриваемая точка есть максимум; возможно, больше двух, если одному и тому же значению y соответствует более одного максимума. (Это не столь уж невероятно, как может показаться. Так, контур буквы W в некоторых типах шрифта может иметь три максимума, соответствующих одному значению y.)

б. Следует ли обращаться к следующей паре границ до окончания работы с текущей парой. Эта ситуация возникает в случае, когда прямая, проходящая через максимум, пересекает одну из текущих границ.

в. Следует ли по достижении конца текущей границы прекратить обращение, поскольку он соответствует минимуму по оси y.

Для примера, приведенного на рис. 8.2, процедура выглядит следующим образом. Сначала считывается первая пара границ *(AB* и *AC)* и соответствующее значение y_{max} — значение любого максимума в этом слое *(D)*. Область, заключенная между границами, в которой значения y превышают значение y_{max}, заполняется (область *B'AC')* и производится обращение к следующей паре границ *(CD* и *DE)*. Теперь мы располагаем текущим списком, содержащим четыре границы; области, заключенные между первой и второй, третьей и четвертой границами, заполнены. Поскольку максимумы точно определены, контроль точности здесь не вызывает затруднений в отличие от контроля четности в процедуре заполнения области на основе анализа значений пикселов (см. разд. 8.3). По окончании работы с некоторой границей из списка, сформированного в результате сортировки, выбирается следующая граница *(EF)* и вводится в текущий список. Однако необходимо выделить случай одновременного окончания двух границ *(AC* и *DC),* когда не нужно вводить в текущий список новые объекты. Этот случай должен быть выделен соответствующей меткой.

В табл. 8 1 приведен список очередности для примера на рис. 8.2. Нижние индексы соответствуют обозначениям точек на рисунке. Значения, помещенные в столбце «Метка (а)», указывают число границ, подлежащих одновременному переносу из спи-

Таблица 8.1

Описание границ для контура, приведенного на рис. 8.2

Сторона	x	y	Δx	Δy	Метка (а)	Метка (б)	Метка (в)
AB	x_A	y_A	$x_B - x_A$	$y_B - y_A$	2	y_D	1
AC	x_A	y_A	$x_C - x_A$	$y_C - y_A$	1	—	0
DC	x_D	y_D	$x_C - x_D$	$y_C - y_D$	2	—	0
DE	x_D	y_D	$x_E - x_D$	$y_E - y_D$	1	—	1
EF	x_E	y_E	$x_F - x_E$	$y_F - y_E$	1	—	1
. . .	. . .	. . .	. . .	. . .	. . .	. . .	. . .

ска очередности в текущий список (число 2 означает, что должны быть перенесены текущая и очередная границы, а число 1 означает перенос только текущей границы). Метка (б) указывает значения y, при которых в текущий описок должна заноситься новая пара границ. Метка (в) указывает число новых границ, к которым должно быть выполнено обращение по окончании работы с текущей границей.

Алгоритм 8.1. **Заполнение области между сторонами многоугольника**

Обозначения. Каждая сторона многоугольника задается координатами y и x своей наивысшей концевой точки, за исключением случая, когда сторона расположена горизонтально. При этом берутся координаты x, y ее левой концевой точки. $\bullet x$ и $\bullet y$ — разности значений координат второй концевой точки прямой и точки, выбранной для задания прямой. Нижний индекс i характеризует исходную индексацию сторон многоугольника. Метки сторон имеют те же значения, что и метки в табл. 8.1.

1. Все стороны сортируются в соответствии со значениями y_i (ранжировка в порядке убывания). В случае равенства значений y_i сортировка проводится в соответствии со значениями x_i, *а* затем — в соответствии с $\bullet x$ и $\bullet y$, причем отрицательные значения предшествуют положительным. {Формирование списка очередности.}
2. For всех пар смежных сторон с одинаковыми y do:
   Begin.
3. Через точку, к которой указанные стороны соприкасаются, проводится прямая и отыскиваются ее точки пересечения со всеми остальными сторонами. {Естественно, необходимо учитывать лишь стороны, расположенные в списке впереди.}
4. For первая сторона, с которой произошло пересечение, помечается в списке очередности значением координаты y точки пересечения.

   End {Список очередности сформирован.}

5. **While** список очередности — непустой **do:**
Begin.
6. If текущий список — пустой **or** значение y меньше соответствующей координаты левой концевой точки одного из элементов этого списка, **then** такие элементы удаляются из списка и из списка очередности в текущий список переносятся новые элементы, число которых определяется метками.
7. Заполнение области, расположенной между чередующимися (четными и нечетными) элементами текущего списка.
8. Уменьшение y.
End.
9. **Конец алгоритма.**

Совершенно очевидно, что в табл. 8.1 содержится избыточная информация. Так, вполне можно обойтись почти без всех значений x_i и y_i (см. задачу 8.2). Мы не затрагиваем здесь проблему кодирования, чтобы не запутывать простой по существу алгоритм. Детальное изложение процедуры заполнения дается в виде алгоритма 8.1. Сортировку, предусмотренную шагом 1, можно выполнять с помощью любого из многочисленных алгоритмов, описанных в литературе. Способ, применяемый для разрешения проблемы равенства, гарантирует размещение в соседних позициях сторон, точка соприкосновения которых образует максимум, причем первой из них идет сторона с положительным угловым коэффициентом. На шаге 2 отыскиваются максимумы. Поскольку значение y_i поставленное в соответствие каждой границе, представляет собой большую координату концевой точки, равенство значений y для двух смежных границ означает наличие максимума. Специфика правила, используемого для разрешения проблемы равенства значений координат при сортировке, позволяет с помощью несложного геометрического доказательства показать, что проверка значений координат x не требуется. Шаги алгоритма 1—4 отводятся на формирование списка очередности и для каждого контура выполняются только один раз. На шагах алгоритма 5—9 происходит собственное заполнение.

Описанный алгоритм не различает контуры отверстий и внешние контуры; его применение не ограничивается односвязными областями. Данный алгоритм обеспечивает правильное заполнение групп областей при условии совместной сортировки их контуров (в частности, обеспечивает заполнение контура буквы *i)*.

Описанный алгоритм не рассчитан на случай, когда частями контура служат горизонтальные прямые, однако его нетрудно модифицировать таким образом, чтобы он выполнял правильную обработку подобных контуров (см. задачу 8.1).

8.3. ЗАПОЛНЕНИЕ КОНТУРА ПО КРИТЕРИЮ ЧЕТНОСТИ

Хотя реализация проверки на четность в алгоритмах заполнения области между сторонами многоугольника тривиальна, она перестает быть таковой при переходе на дискретную плоскость. Помимо сложностей, связанных с выявлением касаний, которые должны учитываться как сдвоенные точки, следует принимать во внимание и кратные пикселы, поскольку они могут исказить результаты подсчета числа пересечений. Трудности, связанные с определением прямых и пересечений прямых и кривых и обсуждавшиеся в гл. 7, должны послужить предупреждением о риске, которым чреват данный подход. В этом разделе будет изложен алгоритм, в котором используется К-ГСС, и затем будет показано, что для полных областей он дает правильные результаты. В алгоритме 8.2 представлены основные операции, связанные с выполнением проверки на четность.

Алгоритм 8.2. Тривиальная процедура проверки на четность на плоскости

0. Задание системы *x*-*y*-координат.
1. **For** каждого *y* do шаги 2—6.
   Begin.
2. Установка *счетчика* на ноль.
3. For каждого *x* слева направо **do** шаги 4 и 5.
   Begin.
4. If *(x, y)* принадлежит контуру, **then** значение *счетчика* увеличивается.
5. **Else if** значение *счетчика* — нечетное, **then** пиксел *(x, y)* находится внутри соответствующей области.
   End.
   End.
6. **Конец алгоритма.**

Совершенно очевидно, что алгоритм 8.2 неточен. Во-первых, в нем ведется подсчет пикселов, а не отрезков, образованных смежными пикселами, которые окрашены в цвет контура. (Напомним, что пересечение двух кривых на дискретной плоскости не обязательно представляется одним пикселом.) Во-вторых, этот алгоритм не предусматривает отыскание экстремумов. Корректный алгоритм, по крайней мере для полных областей, можно построить, если учитывать не пикселы, а отрезки, и учитывать прямые, расположенные над и под прямой, используемой для проверки на четность. Если одна из этих прямых не пересекает контур в окрестности рассматриваемого пересечения, то это означает, что рассматриваемая («проверяемая на четность») прямая является касательной.

Алгоритм 8.3. Алгоритм заполнения области по критерию четности на основе анализа значений пикселов

Обозначения. Значение *счетчика* указывает число пересечений контура рассматриваемой горизонтальной прямой. Переменные *выше* и *ниже* указывают порядки К-ГСС, для определения которых используется процедура *LINK*.

1. For каждого *y* do шаги 2—11.
   Begin.
2. Установка *счетчика* на ноль.
3. Присвоение *x* значения, соответствующего левому краю сетки.
4. **While** *x* меньше или равно значению, соответствующему правому краю сетки **do** шаги 5—11.
   Begin.
5. If точка *(x, y)* не принадлежит контуру, **then do** шаги 6 и 7.
   Begin.
6. If значение *счетчика* — нечетное, **then** *(x, y)* помечается как внутренняя точка области.
7. Увеличение *x*.
   End.
8. **Else do** шаги 9—11.
   Begin.
9. Обращение к **procedure** *LINK*.
10. **If** обе переменные *выше* и *ниже* равны 1, **then** значение *счетчика* увеличивается.
11. **If** сумма переменных *выше* и *ниже* не равна О или 2, **then** ставится признак ошибки.
    End.
    End.
    End.
12. **Конец алгоритма.**

В данном случае в качестве структуры данных удобно использовать граф смежности строк контура (К-ГСС), поскольку его вершины соответствуют пересечениям контура с некоторой горизонтальной прямой. В точке максимума *верхний* порядок вершины равен нулю, а в точке минимума *нижний* порядок вершины равен нулю. Если контур не содержит кратных пикселов, то пересечению дуги с горизонтальной прямой в К-ГСС соответствует вершина с порядками (1,1). Более того, вершины, соответствующие экстремумам, будут иметь порядки (0,2) или (2,0). Любые другие пары значений порядков недопустимы для полной области. Эти условия использованы в алгоритме 8.3. Для определения порядков вершин К-ГСС (посредством подсчета числа интервалов, перекрывающихся с рассматриваемым) в данном алгоритме служит процедура *LINK* (СВЯЗЬ, представленная в виде алгоритма 8.3а). Предполагается, что единицами измерения значений *x* и *y* служат размеры ячеек сетки. Появление признака ошибки в рас-

печатке алгоритма сигнализирует о порядках, отличных от допустимых пар значений (1,1), (0,2) и (2,0). На рис. 8.3 приведено несколько примеров значений порядков вершин ГСС для различных конфигураций пикселов.

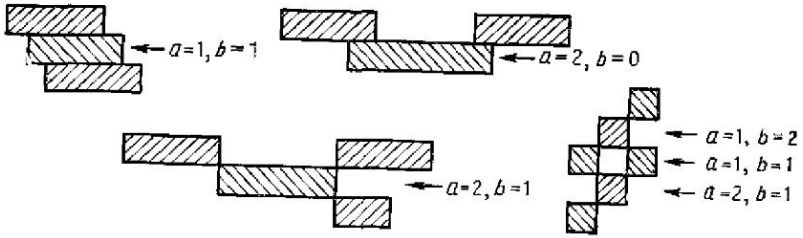


Рис. 8.3. Примеры конфигураций, возникающих при перекрытии интервалов; указанные значения счетчиков использованы в алгоритмах 8.3, 8.3а: символ *а* — *верхний порядок,* символ б — *нижний порядок*

Алгоритм 8.3а. Определение порядка вершины ГСС
Procedure *LINK*

Обозначения, dy — приращение координаты *y* при переходе к очередной строке. Если начало координат находится в левом нижнем углу изображения, то *dy* принимает положительные значения, в остальных случаях — отрицательные, *dx* — приращение координаты *x*. Обычно *dx* и *dy* равны 1.

0. Входной информацией для алгоритма служат координаты *x, y* пиксела, принадлежащего контуру. Выходной информацией являются значения *верхнего* и *нижнего* порядков (переменные *верхний и нижний* соответственно).
1. Установка значений счетчиков *верхний* и *нижний* на ноль.
2. if *(x—dx, y + dy)* принадлежат контуру, then приращение значения *верхний.*
3. If *(x—dx, y—dy)* принадлежит контуру, then приращение значения *нижний.*
4. While *(x, y)* принадлежит контуру do шаги 5—7.
   Begin.
5. If *(x, y + dy)* принадлежит контуру, а *(x—dx, y+dy)* не принадлежит, then приращение значения *верхний.*
6. If *(x, y—dy)* принадлежит контуру, а *(x—dx, y—dy)* не принадлежит, then приращение значения *нижний.*
7. Приращение значения координаты *x.*
   End.
8. If *(x—dx, y+dy)* не принадлежит контуру, а *(x, y+dy)* при надлежит, then приращение значения *верхний.*
9. If *(x—dx, y—dy)* не принадлежит контуру, а *(x, y—dy)* при надлежит, then приращение значения *нижний.*
10. Вывод координат пиксела *(x, y)* и значений счетчиков *верх ний* и *нижний.*
11. **Конец алгоритма.**

8.3.1. ДОКАЗАТЕЛЬСТВО КОРРЕКТНОСТИ АЛГОРИТМА 8.3

Необходимо изучить условия, при которых вводится признак ошибки. Оказывается, это происходит лишь в окрестности кратных пикселов. Поскольку граница представляет собой замкнутую кривую, очевидно, что она пересекает выпуклое множество на плоскости (например, ячейку, соответствующую пикселу) четное число раз. Через обычный пиксел проходит лишь одна подобная дуга и, следовательно, в контуре у этого пиксела должно быть ровно два соседних пиксела. То же справедливо и для множества обычных пикселов, в частности для множества смежных пикселов, расположенных вдоль горизонтальной прямой. С другой стороны, кратный пиксел может иметь в контуре два соседних пиксела или даже один. В таком случае обход контура может оказаться неэквивалентным обходу границы. Ситуация осложняется еще тем, что работа ведется не с отдельными пикселами, а с интервалами. На рис. 8.4 представлен контур, содержащий несколько кратных пикселов; в этом случае использование критерия четности приводит к неверным результатам, что и вызывает появление признака ошибки. В табл. 8.2 приведены значения порядков соответствующих вершин ГСС и те значения *счетчика,* которые следует использовать для правильного заполнения области.

Таблица 82

Интервалы изображения, представленного на рис. 8.4, при обработке которых необходима коррекция

Интервал	Порядки	Правильное значение счетчика
$(H—K)$	(0,3)	0
(A)	(1,2)	0
(D)	(2,2)	2
(B)	(2,1)	1
(L)	(1,2)	1

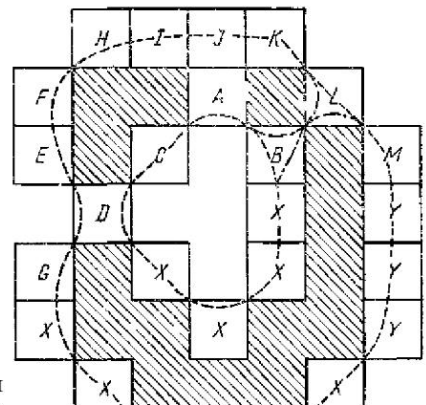


Рис. 8.4. Пример построения границы области при наличии пиксела, имеющего три соседних пиксела; пикселы внутри области заштрихованы

Первоначально может сложиться впечатление, что пикселы B и L не являются кратными в смысле определений, введенных в гл. 7. Можно, однако, одну дугу контура провести через пикселы B, L, K и A, а другую — через пикселы X, B, L, M, Y, ... и т. д. Аналогичный пример неоднозначности приведен на рис. 7.15,*б*; он использовался для обоснования требования о том, что внутренние части полных областей должны быть н-связными. Следует отме-

тить, что кратный пиксел не является необходимым условием возникновения ошибки.

Эти примеры представляют собой скорее иллюстрацию крайнего случая и при решении многих прикладных задач сетка выборки является достаточно точной для того, чтобы исключить появление кратных пикселов. Теперь покажем, что при выполнении этих условий алгоритм 8.3 обладает корректностью.

Теорема 8.1. Если область является полной, то для ее контура допустимы лишь следующие пары значений порядков: (0,2), (2,0) и (1,1).

Доказательство. Докажем эту теорему, продемонстрировав невозможность появления иных пар значений. Рассмотрим пару значений (0,1). Если данный интервал и интервал, расположенный под ним, содержат по одному пикселу, то имеем дело с пикселом, кососеди которого совпадают (рис. 8.5,*а*). Если интервал, расположенный ниже рассматриваемого, содержит два пиксела, то они являются н-соседями друг относительно друга (рис. 8.5,*б*).

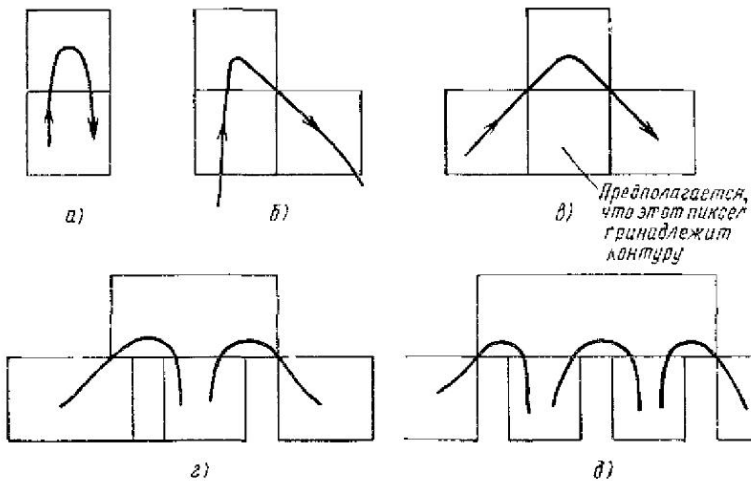


Рис. 8.5. Примеры, использованные при доказательстве теоремы 8.1:
а — значения порядков (0,1) — один пиксел на строке, расположенной ниже рассматриваемой, б — значения порядков (01) — два пиксела на строке, расположенной ниже рассматриваемой, в — значения порядков (0,1) — три пиксела на строке, расположенной ниже рассматриваемой; г — значения порядков (0,3), *д* — значения порядков (0,4)

Если в нем содержится более двух пикселов, то некоторые из них являются н-соседями пикселов контура, содержащихся в строке, расположенной над ними (рис. 8.5,*в*). Возникновение двух последних случаев не связано с числом пикселов, содержащихся в рассматриваемом интервале. Все три перечисленные конфигурации не соответствуют условиям определения 7.6 и, следовательно, не могут встречаться при полной области. Случай пары значений порядков (1,0) можно проанализировать аналогичным образом. Теперь необходимо показать, что при полной области ни одно зна-

чение порядка не может превышать 2. Если значение порядка — нечетное, то это означает, что одна из ветвей обходится многократно (рис. 8.5,*г*), Если значение порядка — четное, то у некоторых пикселов на маршруте должно иметься более двух н-соседей (рис. 8.5,*д*).

Следствие. Если для обработки контура полной области используется алгоритм 8.3, то признак ошибки не появляется ни при каких обстоятельствах (шаг 11).

Итак, для заполнения контуров полных областей можно использовать сравнительно простой алгоритм, который одновременно работает с малым числом строк. Если область — неполная, то такой алгоритм может ошибаться, что было показано на рис. 8.4 и в табл. 8.2. К сожалению, при работе с неполными областями этот алгоритм не только допускает ошибки, но и не выводит признак ошибки. На рис. 8.6 представлен соответствующий пример (признак ошибки не появляется до тех пор, пока не будет

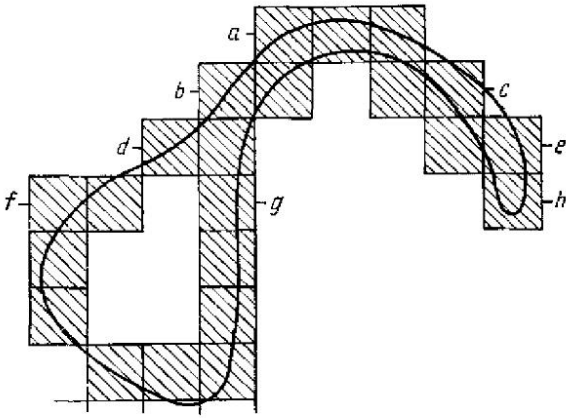


Рис. 8.6 Пример контура, правильное **заполнение которого с** помощью алгоритма ***S3* невозможно**

проведено какое-либо неверное заполнение: пиксел, лежащий между интервалами *b* и *c)*. В этом случае верхняя вершина характеризуется порядками (0,2), в то время как соответствующий интервал содержит четыре дуги границы. Отметим, что единственный способ правильно заполнить такой контур с помощью алгоритма заполнения области по критерию четности на основе анализа значений пикселов состоит в более чем однократном обходе контура. Правильное заполнение произвольной области *R* достижимо при условии, что ее контур рассматривается как область и его обход осуществляется с целью отыскания нового контура, в процессе которого пикселы снабжаются метками, указывающими их кратность (см. разд. 8,6).

8.3.2. РЕАЛИЗАЦИЯ АЛГОРИТМА ЗАПОЛНЕНИЯ ОБЛАСТИ ПО КРИТЕРИЮ ЧЕТНОСТИ

Заполнение области по критерию четности на основе анализа пикселов имеет определенные преимущества перед заполнением между сторонами многоугольника, поскольку как сортировка по значениям координаты *y,* так и интерполяция между заданными точками (если в значениях *y* имеются разрывы) выполняются не на главной ЭВМ, а на дисплее. Сначала можно воспроизвести контур с помощью команды *vec(x, y)* (см. разд. 1.7, табл. 1.2). В этом случае интерполяция осуществляется аппаратной частью устройства воспроизведения без затрат ресурсов главной ЭВМ. При этом обеспечивается правильность сортировки точек, поскольку процесс воспроизведения изображения можно рассматривать как «сортировку участков записей». Здесь открываются две возможности. С одной стороны, соответствующий алгоритм можно реализовать в микропроцессоре, управляющем дисплеем; при этом сеткой является память обновления изображения. С другой, можно с помощью команды *rpic* (см. разд. 1.7, табл. 1.2) ввести в память главной ЭВМ группу, состоящую из трех строк. В ней проводится проверка четности, после чего средняя строка возвращается обратно. Платить за это удобство приходится тем, что только полные области заполняются правильно.

8.4. ЗАПОЛНЕНИЕ КОНТУРА ПО КРИТЕРИЮ СВЯЗНОСТИ

Если известно, что точка принадлежит внутренней части области, то ее можно использовать в качестве своего рода *затравочного пиксела,* окрашивая в ее цвет смежные пикселы до тех пор, пока не встретится контур. Если для этого алгоритма воспользоваться рекурсивным описанием, то выглядит оно достаточно просто. Пусть *c* цвет затравочного пиксела 5 и *F* — заполняемая область, степень заполнения которой соответствует шагу алгоритма. На первом шаге задается F={S}, затем на каждом шаге к области *F* добавляются все те пикселы, которые окрашены в цвет *c* и имеют непосредственного соседа в *F.* Таким образом, заполнение контура осуществляется на основе связности. Для обозначения этого процесса используют и название «алгоритм засеивания». Допущение, что имеется такой затравочный пиксел, при работе на интерактивных графических системах является вполне реалистичным, поскольку в этом случае пользователь может указать этот пиксел, воспользовавшись световым пером или введя световой указатель внутрь контура. То же относится и к системам, в которых трехмерный объект задается вершинами аппроксимирующего его многогранника и на экране отображаются различные проекции этого объекта. При этом нетрудно определить точки, расположенные внутри получающихся в результате такого отображения многоугольников. В системах обработки

изображений точку, расположенную внутри области, можно найти, взяв точку, непосредственно смежную с пикселом контура, и снабдив *ее* меткой в процессе выделения контура. В качестве средства предварительной обработки можно использовать и частичную проверку на четность.

При заполнении области по критерию связности появление в области «паразитных» пикселов менее вероятно, чем при заполнении по критерию четности. Действительно, цвет внутренней части рассматриваемой зоны области известен и пиксел не вводится в область, если он окрашен в другой цвет. Паразитный пиксел может появиться лишь тогда, когда заданный контур не замкнут. Гарантия отсутствия паразитных пикселов оплачивается необходимостью иметь по одному затравочному пикселу на каждую компоненту связности области в тех случаях, когда в области содержится более одной такой компоненты.

8.4.1. РЕКУРСИВНАЯ ПРОЦЕДУРА ЗАПОЛНЕНИЯ ПО КРИТЕРИЮ СВЯЗНОСТИ

Обход внутренней части области можно осуществлять различными способами с помощью структур данных, рассмотренных в гл. 6. Алгоритм, простейший в смысле длины соответствующей программы, основан на рекурсии и использует сетку пикселов. Он представлен ниже в виде алгоритма 8.4. В нем предусматривается обследование четырех н-соседей затравочного пиксела и к каждому из них, не принадлежащему контуру, снова применяется процедура *FILL* (ЗАПОЛНЕНИЕ), при выполнении которой затравочным пикселом служит уже этот сосед. С помощью небольшой модификации процедуры можно вводить во внутреннюю часть области любой пиксел, окрашенный в тот же цвет, что и затравочный пиксел.

Алгоритм 8.4. Рекурсивное заполнение контура

Обозначения. Затравочный пиксел — заданный пиксел, C — цвет внутренней части заполняемой области. Пикселы контура уже окрашены в этот цвет. Переменная p_N обозначает N-соседа пиксела p, $Q<=N<=7$ (см. определение 7.1).

1. **Call** *FILL (затравочный пиксел).*
2. **Конец алгоритма.**

Процедура *FILL (p)*

F1. Присвоение C значения цвета, в который окрашен пиксел p.
F2. **For** $N=0, 2, 4,$ 6 do:
Begin.
F3. **If** цвет пиксела P_N отличается от C, **then call** *FILL* (p_N).
End.
F4. **Return.**

Простота этого алгоритма только кажущаяся, так как каждое обращение к подпрограмме сопровождается определенными зат-

ратами вычислительных ресурсов, размер которых трудно оценить по записи алгоритма. В разд. 8.5 мы еще вернемся к этой проблеме.

8.4.2. НЕРЕКУРСИВНАЯ ПРОЦЕДУРА ЗАПОЛНЕНИЯ ПО КРИТЕРИЮ СВЯЗНОСТИ

Вместо того чтобы возлагать весь контроль за процессом обхода на систему, управляющую вызовом подпрограммы, мы можем сами заняться этим, причем с большей эффективностью. При работе с растровыми графическими устройствами особенно удобно пользоваться ГСС. В простейшем варианте алгоритм начинает работу с затравочного пиксела и «перемещается» влево до тех пор, пока не будет обнаружен пиксел контура. После этого осуществляется просмотр вправо снова вплоть до контура. Далее заполняется следующая строка, и процесс продолжается до тех пор, пока не будет достигнута самая нижняя часть контура. Эта процедура повторяется для той части области, которая расположена выше начального пиксела. Процесс достаточно прост, однако нетрудно убедиться в том, что он обеспечивает правильное заполнение контура лишь при условии выпуклости соответствующей области. Данную процедуру можно усовершенствовать, чтобы она обеспечивала заполнение невыпуклых областей — необходимо предусмотреть стек, в котором будут храниться пикселы, расположенные в окрестностях максимумов и минимумов контура. Впоследствии эти пикселы выталкиваются из стека и используются в качестве затравочных. Так, в примере, приведенном на рис. 8.7, пикселы *A, B я C* будут помещены в стек при заполнении заштрихованной части области. Пикселы *D* и *E* будут помещены в стек позже — при обходе незаштрихованной части области. Хотя сама по себе эта идея очень проста, ее правильная реализация требует определенной тщательности.

Прежде чем перейти к описанию основного алгоритма, остановимся на некоторых процедурах, которые в нем используются.

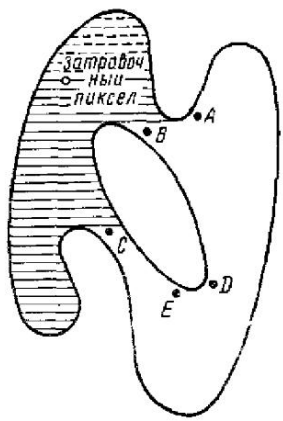


Рис. 8.7. Заполнение невыпуклой области по критерию связности.

Используя в качестве исходной точки заданный затравочный пиксел, алгоритм продвигается вниз, заполняя часть области, заштрихованную сплошными линиями; затем алгоритм возвращается к затравочному пикселу и начинает продвижение вверх, заполняя часть области, отмеченную штриховыми линиями

8.4.3. ПРОЦЕДУРЫ, ИСПОЛЬЗУЕМЫЕ ПРИ ЗАПОЛНЕНИИ ПО КРИТЕРИЮ СВЯЗНОСТИ

Мы будем пользоваться термином «адрес пиксела» для обозначения описания местоположения соответствующего пиксела В роли такого описания могут выступать координаты *x, y,* указатель массива и т. п. Если *p* — адрес пиксела *A,* то выражение *p—1* задает адрес пиксела, расположенного слева от пиксела Л (на той же горизонтальной прямой), а выражение *p*+1 — адрес пиксела, расположенного справа от пиксела *A*. Первый из этих адресов не определен для крайнего правого, а второй — для крайнего левого пикселов каждой строки. При любой реализации таких алгоритмов необходимо предусматривать специальные средства для работы с этими пикселами, однако мы не будем усложнять изложение рассмотрением подобных частных случаев.

Процедура *LEFT(p)* определяет адрес крайнего левого пиксела интервала с фиксированным цветом, в который входит пиксел *p* (рис. 8.8,*а*). Процедура *LRIGHT(p)* состоит в следующем:

LRIGHT (p) = LEFT (LEFT (p) - 1) - 1,

т. е. она определяет адрес такого крайнего правого пиксела, расположенного слева от пиксела *p,* который окрашен в тот же, что и пиксел *p,* цвет и отделен от него, по меньшей мере, одним пикселом другого цвета (рис. 8.8,*б*). Реализация этих процедур очевидна.

Порядки вершины (интервала) ГСС, соответствующей пикселу *p,* определяются с помощью процедуры *LINK(p),* описанной в виде алгоритма 8.3а. Для определения адресов пикселов e_1 и e_2, представленных на рис. 8.8,*в*, требуется небольшая модификация этой процедуры. Если такие пикселы не заданы, процедура вместо адреса выдает ноль. (Если ноль — допустимый адрес, то необходимо выбрать в качестве начальных значений e_1 и e_2 в процедуре *LINK,* какие-либо другие значения.)

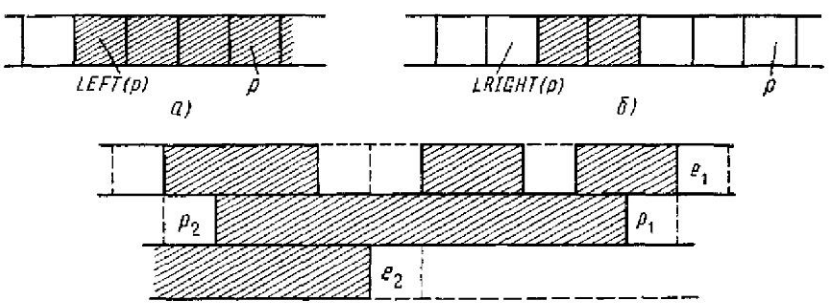


Рис. 8.8. Определение процедуры *LEFT(p)* *(а)* и процедуры *LRIGHT (б);* задание пикселов p_1, p_2, e_1 и e_2 *(в).*

Аргументом *p* процедуры *LINK* может служить любой пиксел, находящийся на заштрихованном участке между пикселами p_2 и p_1

Поскольку затравочный пиксел может находиться в середине области, необходимо вести ее заполнение в двух направлениях, как показано на рис. 8.7. Для всякого другого пиксела, хранящегося в сетке, необходимость подобного обхода в двух направлениях отсутствует. Вместо того чтобы предусматривать специальную процедуру, для обхода первой области целесообразно обратить внимание на задание начальных условий. Если пиксел со значением *затравочный пиксел*$_{верхний}$ (расположен непосредственно над затравочным кристаллом) не принадлежит контуру, то пиксел,адрес которого определяется с помощью процедуры *LEFT (затравочный пиксел*$_{верхний}$*),* следует ввести в стек, снабдив его меткой, указывающей, что с него должен начинаться просмотр вверх. Это, очевидно, достаточно стандартная ситуация при работе с интерактивными графическими системами, поскольку пользователь часто склонен задавать затравочный пиксел подальше от контура. Если пиксел, расположенный над затравочным пикселом, принадлежит контуру, процесс заполнения осуществляется следующим образом. Сначала с помощью обращения к процедуре *LINK* определяются порядки соответствующего интервала контура. Если оказывается, что порядки имеют значения (0,2), то это означает, что затравочный пиксел размещается непосредственно под точкой максимума и, следовательно, отсутствует необходимость в просмотре вверх. В противном случае затравочный пиксел размещается непосредственно под почти горизонтальной дугой контура. Затем затравочный пиксел заменяется пикселом с адресом *LEFT (затравочный пиксел)* и производится обращение к процедуре *LINK,* которая применяется к интервалу контура, расположенному слева от затравочного пиксела. После этого пиксел e_1 назначается затравочным пикселом для просмотра области вверх.

8.4.4. ОПИСАНИЕ ОСНОВНОГО АЛГОРИТМА

Содержательно этот алгоритм можно описать следующим образом. Горизонтальные строки заполняются в процессе обхода слева направо, начиная с пиксела, расположенного непосредственно справа от контура. [Для отыскания последнего в первой строке можно воспользоваться процедурой *LEFT (затравочный пиксел).]* Пусть p — такой затравочный пиксел. Прежде чем приступать к собственно заполнению, производится обращение к процедуре *LINK*(p—1) с тем, чтобы проанализировать К-ГСС левого интервала контура. Если значения *верхнего* и *нижнего* порядков=1, то интервал представляет собой простую дугу и остается определить положение следующей просматриваемой строки. Если направление вертикального просмотра имеет значение «вниз», то задается $p_{след} = e_2$, в противном случае $p_{след} = e_1$. Далее продолжается процесс заполнения по горизонтали, начиная с пиксела p, вплоть до обнаружения пиксела, принадлежащего контуру. Пусть $P_{прав}$ — адрес последнего пиксела, находящегося внутри области. Производится обращение к процедуре *LINK*($p_{прав}$+1). Если в

результате ее выполнения выясняется, что значения *верхнего* и *нижнего* порядков равны 1, то пиксел p заменяется пикселом $p_{след}$ и весь описанный процесс повторяется.

Интерес представляют случаи, когда на обоих концах равно 1 только одно из значений порядков. Если, в частности, одно из них равно нулю, то это означает, что в соответствующей точке находится некоторый экстремум и следует либо поместить пиксел в стек, либо прекратить заполнение в данном направлении. Это условие составляет существенную часть рассматриваемого ниже алгоритма.

Описанные механизмы представляют собой основу алгоритма 8.5. В нем помимо стека 5 используется стек S_d, в который одновременно с вводом адреса пиксела в стек 5 заносится соответствующее направление просмотра. (Допустимо, естественно, использовать и общий стек, элементами которого служат пары значений — адрес пиксела и направление просмотра.) Процедура *POP (стек)* описана в алгоритме 6.1. На шаге 3 некоторым переменным присваиваются значения в соответствии с текущим направлением просмотра. В частности, при $u = 1$ в качестве следующего выбирается пиксел e_1, а при $u = 2$ — пиксел e_2. Использование этих переменных избавляет от необходимости иметь отдельные коды для просмотра вверх и вниз. Основную часть алгоритма образует цикл, включающий шаги 5—17. Чтобы просмотреть верхнюю часть изображения, затравочный пиксел приходится помещать в стек дважды.

На шаге 5 определяется, является ли адрес следующего пиксела допустимым или этот пиксел уже прошел заполнение. В обоих случаях алгоритм выходит из внутреннего цикла и выполняет шаг 1. Если стек — непустой, выталкивается новый пиксел, после чего весь процесс повторяется. Если адрес следующего пиксела является допустимым и этот пиксел еще не проходил заполнения, то этот пиксел выбирается в качестве текущего (шаг 6). На шаге 7 определяются порядки К-ГСС для левой части контура.

На шаге 8 выполняется первая нетривиальная операция: проверяется достижение границы внутренней части области в текущем направлении просмотра. Если просмотр осуществляется вниз и значение *верхнего* порядка превышает единицу, то значит процесс заполнения дошел до «дна» контура (как показано на рис. 8.9,*а*) или возникла одна из представленных на рис. 8.9,*б* конфигураций. Наличие первой ситуации определяется посредством проверки того, находится ли текущий пиксел p справа от пиксела $p_{прав}$, т. е. вне контура. В таком случае заполнение в этом направлении прекращается и алгоритм выходит из данного цикла. В противном случае текущий пиксел действительно находится внутри области. (При просмотре вверх используется аналогичный тест Для определения достижения «верхушки» контура.)

На шаге 9 проверяется, не равны ли нулю *верхний* и *нижний* порядки, и выбирается начальный пиксел в следующей строке. Не обязательно, чтобы оба значения порядков равнялись единице,

даже несмотря на то, что вершины со значениями порядков (1,2) и (2,1) для полных областей недопустимы. Для большинства пикселов контура условия этого шага выполняются и алгоритм переходит непосредственно к шагу 14. Если значение хотя бы одного из порядков равно нулю, то значит соответствующий пиксел следует поместить в стек (см. утверждение 8.1) и при выборе пиксела $p_{след}$ необходимо проявить определенную осмотрительность.

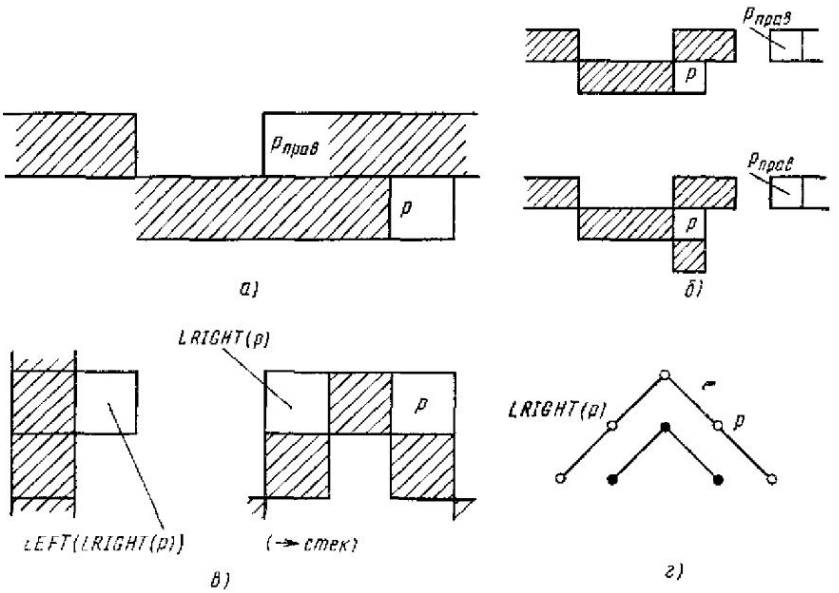


Рис. 8.9. Иллюстрация к работе алгоритма 8.5:
а, б — конфигурации, для которых условие шага 8, заключенное в квадратные скобки ([...]), выполняется при просмотре вниз. Достижение «дна» контура и прекращение просмотра в данном направлении зафиксировано лишь для конфигурации *а*; *в* — при обнаружении представленной конфигурации в процессе осмотра левой части контура пиксел *p* вводится в стек; *г* — два ГСС для конфигурации *в*

Ввод пиксела в стек осуществляется на шаге 10. На рис. 8.9,0 представлена конфигурация пикселов, при которой значение *верхнего* порядка равно нулю. Отметим, что наличие дуги контура, изображенной на рисунке слева, гарантировано, если контур действительно замкнут. В противном случае пикселы, расположенные на строке, находящейся над пикселом *p*, будут окрашены в тот же цвет, что и пиксел *p*, и не будут ограничены слева. На рис. 8.9,г для этого случая представлены К-ГСС (вершины обозначены зачерненными кружками) и В-ГСС (вершины обозначены незачерненными кружками).

Появление нулевого порядка означает, что начальный пиксел на очередной строке просмотра следует выбирать особенно тщательно. Если просмотр осуществляется вниз и значение *верхнего* порядка равно, а *нижнего* не равно нулю, то в этом случае никаких сложностей не возникает. Эта проверка, как и аналогичная

проверка для случая просмотра вверх, осуществляется на шаге 11. На рис. 8.10,а изображена конфигурация контура, для которой выполняются условия шага 11 для случая просмотра вниз.

Шаг 12 рассчитан на случай, когда в следующей строке (по направлению просмотра), примыкающей к контуру, отсутствуют пикселы контура. Такая конфигурация изображена на рис. 8.10,*б*

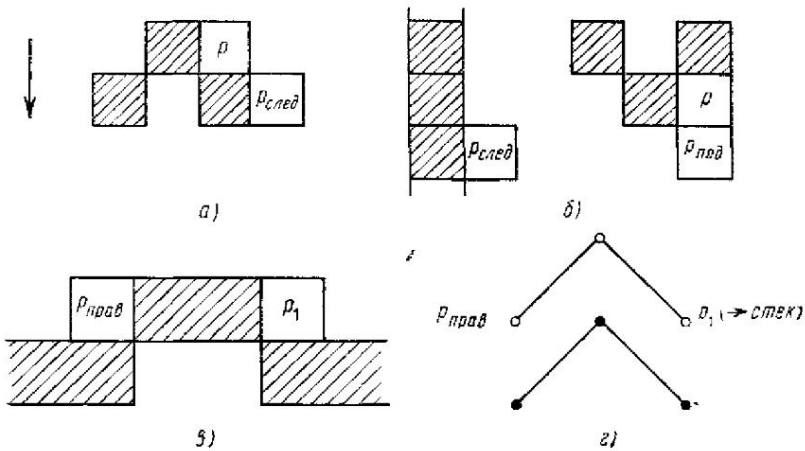


Рис. 8.10. Иллюстрация к работе алгоритма 8.5:
а — конфигурация, для которой выполняются условия шага 11, *б* — шага 12, *в* — если при исследовании правой части контура возникает представленная конфигурация, пиксел *p* заносится в стек; *г* — два ГСС для конфигурации в

для случая просмотра вниз. Рассматривается пиксел, расположенный непосредственно над ($p_{над}$) или непосредственно под ($p_{под}$) текущим пикселом. Если рассматриваемый пиксел не прошел заполнения, то он выбирается в качестве следующего пиксела на линии просмотра, расположенной непосредственно справа от контура (см. комментарии к шагу 10, касающиеся существования подобной дуги контура).

На шаге 14 осуществляется реальное заполнение. Именно на этом шаге осматривается большинство пикселов. При его выполнении для каждого пиксела проверяется, не окрашен ли он в цвет контура и не заполняет ли он пиксел с правильным цветом.

Алгоритм 8.5. Заполнение контура по критерию связности

Обозначения. $P_{след}$ — пиксел, с которого начинается заполнение при переходе к новой линии просмотра. $p_{др}$ — пиксел, расположенный непосредственно над или под текущим пикселом; может принимать значения $p_{над}$ и $p_{под}$. S — стек, в котором содержатся адреса затравочных пикселов; S_d — стек, в котором хранятся значения направлений обхода ГСС. 0. В качестве входной информации задается массив, представляющий изображение, адрес *затравочного* пиксела, расположенного внутри заполняемой области, и иногда еще один *зат-*

равочный пиксел$_{над}$. Адрес *LEFT (затравочный пиксел)* заносится в стек S, а значение направления просмотра *вниз* — в стек S_d. Если задан второй пиксел, его адрес *LEFT (затравочный пиксел*$_{над}$*)* также заносится в стек S, а направление просмотра *вверх* — в стек S_d.

1. **While** стек 5 — непустой, **repeat** шаги 2—19.

Begin.

2. $p_{след}$ = *POP(S), направление = POP* (S_d), $p_{прав}=X_{max}$.
3. If *направление* равно *вниз,* **then** присваиваются $и$ = 2, *др = под.* **Else** $и$=1, *др — над,*
4. Repeat шаги 5—16

Begin.

5. If $p_{след}$ равен 0 or if $p_{след}$ прошел заполнение, **then exit** из цикла.
6. $p=p_{след}$
7. *LINK (p*—1) {исследование левой части контура}
8. If *[(направление* равно *вниз* и значение *верхнего* порядка больше 1) или *(направление* равно *вверх* и значение *нижнего* порядка больше 1)] и пиксел p расположен справа от пиксела $p_{прав}$, **then exit** из цикла.
9. If значения *верхнего* и *нижнего* порядков больше O, **then** задается $p_{след} = e_и$

Else do:

Begin.

10. If пиксел p ещё не прошел заполнение, then в стек S заносится адрес *LEFT (LRIGHT(p)),* а в стек S_d — значение *направления.*
11. If (значение *верхнего* порядка равно 0, а значение *нижнего* порядка не равно 0, значение *направления* равно *вниз)* or (значение *нижнего* порядка равно 0, а значение *верхнего* порядка не равно 0, значение *направления* равно *вверх),* **then** задается $p_{след}$= $= e_и$.
12. **Else** if рар не прошел заполнение, then задается $p_{след} = LEFT(p_{др})$.
13. **Else** задается $p_{след}$ = 0.

End.

14. Заполнение строки, начиная с пиксела p; пусть пиксел $p_{прав}$ является последним пикселом перед контуром.
15. *LINK*($p_{прав}$+1) {исследование правой части кон тура}
16. If значение *верхнего* или *нижнего* порядков равно O и if пиксел p_1 не прошел заполнения, then **do:**

Begin.

17. Адрес пиксела p_1 заносится в стек S.

18. If значение *верхнего* порядка больше О, then значение направления *вверх* заносится в стек S_d.
19. Else значения направления *вниз* заносятся в стек S_d.
End.
End.
End.
20. Конец алгоритма.

На шаге 15 определяются порядки вершины К-ГСС, соответствующей правой дуге контура. Если значение одного из ее порядков равно 0, то адрес пиксела, расположенного по другую сторону контура (пиксел P_1 на рис. 8.10,*в*), заносится в стек; в стек заносится также соответствующее значение направления. (Шаги 15—17.) Два ГСС, представляющие эту конфигурацию, приведены на рис. 8.10,*г*.

Очевидно, что одновременно в стеке находится максимум две вершины. С другой стороны, все вершины, связанные с заданной вершиной, либо попадут в стек, либо будут заполнены. Рассмотрим пример, приведенный на рис. 8.11. Пусть C — первая вершина, пройденная при обходе. В этом случае вершины B и D *заносятся* в стек. При каждом повторном проходе через вершину D (либо в результате ее выталкивания из стека, либо при заходе в нее из другой вершины) вершина E будет помещаться в стек.

Можно показать также, что в случае полной области алгоритм 8.5 обеспечивает корректное заполнение. В самом деле, при этом в стек заносятся лишь те пикселы, которые являются смежными точками максимумов или минимумов по вертикали.

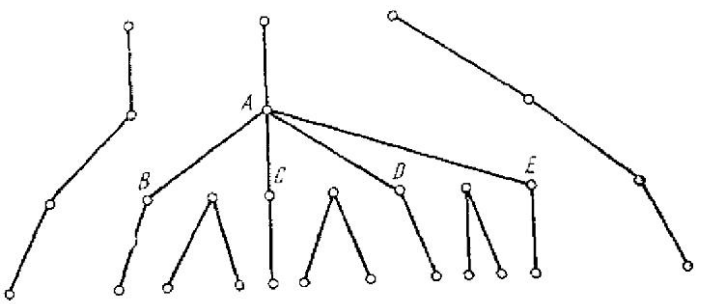


Рис. 8.11 Пример использования стека в алгоритме 8.5

8.5. СРАВНЕНИЕ И КОМБИНАЦИИ АЛГОРИТМОВ

Если сравнивать алгоритмы заполнения, основанные на анализе значений пикселов, используя в качестве критерия длину реализующей их программы, то простейшим явно окажется алгоритм 8.4, следующим будет алгоритм заполнения по критерию четности, алгоритм 8.3, и последним — алгоритм 8.5. Вычислительные процедуры, используемые в алгоритмах 8.3 и 8.5, в некоторых от-

ношениях одинаковы. Оба алгоритма опираются на одни и те же структуру данных и основную вычислительную процедуру. Они обращаются к подпрограмме только при работе с пикселами контура, так что число таких обращений приблизительно равно числу горизонтальных интервалов контура (другими словами, числу вершин К-ГСС). Пикселы, находящиеся во внутренней части заполняемой области, имеют последовательную адресацию, причем для каждого из них проводится только одна простая проверка — выясняется принадлежит пиксел контуру или нет. Последовательную адресацию можно очень эффективно реализовать на большинстве ЭВМ с помощью *индексных регистров.* Рекурсивный алгоритм использует при обработке каждого пиксела внутренней части области до четырех обращений к подпрограмме. Эти обращения являются дорогостоящими, что связано с необходимостью сохранять содержимое регистров, а также информацию, требующуюся для возвращения в начальную точку. Последнюю операцию можно эффективно выполнять на ЭВМ, в которых стеки реализованы аппаратно. Таким образом, основные затраты, связанные с использованием рекурсивного алгоритма, определяются необходимостью сохранять содержимое ячеек памяти (в случае, если соответствующие регистры используются как вызывающей, так и вызываемой программами). При использовании языка высокого уровня эти затраты довольно велики, однако можно попробовать написать программу, реализующую этот алгоритм, на языке ассемблера, для которого указанные затраты являются умеренными (см. задачу 8.5).

Теперь сравним между собой два нерекурсивных алгоритма. Заполнение области по критерию связности обладает двумя основными преимуществами: при связном контуре гарантируется отсутствие «утечки» пикселов; время вычислений пропорционально площади заполняемой области, а не площади прямоугольника, описывающего заполняемый контур, как это имеет место при использовании алгоритмов заполнения по критерию четности. Основные недостатки заполнения области по критерию связности состоят в следующем: а) необходимость знать хотя бы одну точку, лежащую во внутренней части области — при решении некоторых прикладных задач отыскание такой точки может вызвать затруднения; б) если заполняемая область имеет связный контур, но ее внутренняя часть не является связной, то заполненной окажется лишь та часть области, в которой находится затравочный пиксел (алгоритм заполнения по критерию четности обеспечит заполнение всех частей области); в) обход внутренней части области не соответствует организации растра, что может приводить к чрезмерному росту объема обмена между главной ЭВМ и управляющим устройством графического дисплея. Этот недостаток приобретает значение в связи с соображениями, приведенными в подразд. 8.3.2. Если нет возможности запрограммировать соответствующим образом микропроцессор, то все горизонтальные сроки можно считывать последовательно, размещая в оперативной па-

мяти три строки одновременно и используя их для определения порядков вершин ГСС. При использовании контроля четности строка, просмотренная один раз, больше просмотру подвергаться не должна. В случае алгоритмов заполнения по критерию связности дела обстоят уже не так. (Именно поэтому при заполнении приходится использовать значение яркости, отличное от значения яркости контура.) Итак, реальные вычислительные затраты могут оказаться выше из-за значительного объема операций ввода-вывода.

Поскольку в алгоритмах обоих типов основной является процедура *LINK,* можно построить некоторый комбинированный алгоритм, который одни части изображения будет заполнять по критерию четности, а другие с помощью процесса засеивания. Такой метод может сыграть существенную роль при решении тех прикладных задач, в которых заранее трудно задать затравочный пиксел. Реализовать этот метод можно с помощью алгоритма 8.3, если отказаться от заполнения интервалов и не обращать внимание на появление признака ошибки. Если при обработке некоторой строки обнаруживается ошибка, то производится выбор некоторого затравочного пиксела. Мы можем ослабить ограничения, налагаемые на значения порядков, потребовав, чтобы они были больше единицы или равны ей, отказавшись от условия равенства их значений единице. (Эта ослабленная форма ограничений была использована при построении примеров, приведенных на рис. 8.12 и 8.13.) В этом случае заполнение может осуществляться по критерию связности для того, чтобы избежать необходимости обхода изображения вне пределов контура заполняемой области.

С другой стороны, в качестве основного алгоритма можно выбрать процедуру заполнения по критерию четности и обращаться к критерию связности в случае обнаружения таких строк, при работе с которыми определение принадлежности пикселов внутренней части области с помощью критерия четности вызывает затруднения. Такой алгоритм не пропустит при заполнении те внутренние части области, которые не соединены с частью, содержащей затравочный пиксел.

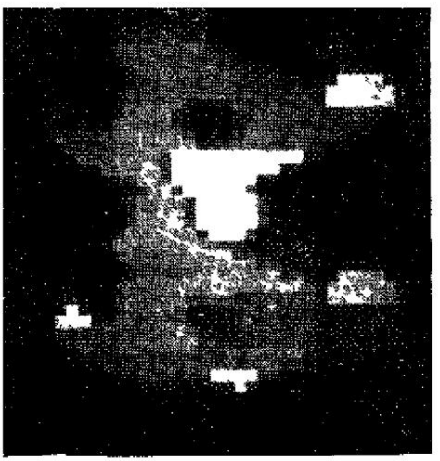


Рис. 8.12. Последовательность заполнения отдельных зон внутренней части области

Имеет смысл отметить, что большая часть вычислений, предусматриваемых алгоритмом 8.5, приходится на окрестности вертикальных экстремумов контура, т. е. на те точки, число которых довольно незначительно. На рис. 8.12 приведен пример области,

заполненной с помощью алгоритма 8.5, который выбирает новое значение уровня серого тона при каждом появлении некоторого пиксела из стека. Здесь предварительная обработки проводится с помощью алгоритма заполнения по критерию четности, который



Рис. 8.13. *Слева:* контур (обозначенный единицами), подвергнутый однородному заполнению. *Справа:* контур и область, ограниченная им, заменены текстом: промежутки между словами представлены символами подчеркивания (-),
На рисунке воспроизведен следующий текст (дается перевод без сохранения формы представления текста) Задача заполнения контура области некоторым заданным цветом решается двумя методами Первый предусматривает проведение горизонтальных прямых пересекающих контур, и закрашивание пикселов, расположенных справа от точек пересечения с нечетными порядковыми номерами (при условии, что левый конец прямой находится)

находит затравочный пиксел в левой верхней части внутренней области. После этого заполнение осуществляется на основе критерия связности. Каждое значение уровня серого тона соответствует некоторому пикселу, вытолкнутому из стека. Алгоритм легко приспособить для заполнения изображения таким образом, что цвет заполняемых пикселов выбирается в зависимости от их адресов (вместо заполнения пикселами одного цвета) Соответствующий пример приведен на рис. 8 13.

8.6. БИБЛИОГРАФИЧЕСКИЕ ЗАМЕЧАНИЯ

Заполнение области между сторонами многоугольника является, вероятно, первым из предложенных и наиболее широко используемым в машинной графике алгоритмом заполнения. Совершенно очевидно, что им целесообразно пользоваться при решении тех прикладных задач, в которых один и тот же контур приходится воспроизводить многократно, как, например, в фотонаборе. Еще одной прикладной областью, в которой уместно применение метода заполнения между сторонами многоугольника, является

плоская мультипликация (синтез на ЭВМ двухмерных динамических изображений), в которой контуры могут переноситься, но не поворачиваться. Алгоритм 8.1 представляет собой вариант алгоритма *YX,* описанного в монографии [1.4] и модифицированного на основе анализа работы алгоритмов, используемых в фотонаборных устройствах. Большая часть «усилий» этих алгоритмов затрачивается на получение адекватного кодового представления контура. Их применение может оказаться чрезвычайно неэффективным при решении таких прикладных задач, в которых некоторый заданный контур будет заполняться очень небольшое число раз. В этих случаях, особенно при работе на растровых графических устройствах, лучше пользоваться алгоритмами заполнения области на основе анализа значений пикселов. Отказ от аналоговой топологической информации, характеризующей контур, не может пройти бесследно, хотя при решении некоторых прикладных задач эта информация всегда остается недоступной. Ряд работ посвящен методам решения данной задачи в случаях, когда контур задается в виде некоторого набора неупорядоченных пикселов В статьях [81, 83, 8.4] предлагаются модифицированные варианты заполнения на основе критерия четности. В последней работе описано обобщение алгоритма 8.3, позволяющее корректно осуществлять заполнение неполных областей при условии, что не существует пикселов, через которые проходит более двух дуг контура либо проходят две дуги контура, одна из которых содержит экстремум по координате *y.* Алгоритмы заполнения по критерию связности описаны в работах [6.10, 82, 8.5]. Алгоритм, представленный в работе (8.2], не обладает корректностью, что и было отмечено в работе [6.10], в которой изложен корректный алгоритм, реализующий, в сущности, обход В-ГСС.

8.7. ЗАДАЧИ

8.1. Алгоритм заполнения области между сторонами многоугольника, рассмотренный в разд. 8.2, не приспособлен для работы с горизонтальными границами. Видоизмените его таким образом, чтобы объекты, имеющие такие границы, обрабатывались этим алгоритмом правильно

8 2. Предложите структуру для списка очередности, в котором должны храниться лишь значения • *x* и • *y* и значения координаты *y* в точках экстремума, любой другой информации в нем должно быть как можно меньше.

8.3 Видоизмените алгоритм 8 За таким образом, чтобы он обеспечивал вычисление адресов пикселов e_1 и e_2, используемых в алгоритме 8 5.

8.4 Оцените затраты на заполнение окружности при использовании алгоритма заполнения области между сторонами многоугольника, алгоритма заполнения по критерию четности и алгоритма заполнения по критерию связности. Получите такие же оценки для случая заполнения кругового кольца.

8 5. (При решении этой задачи предполагается, что читатель знаком с языком ассемблера) Предложите реализацию алгоритма 8 4, в которой регистры вообще не используются для хранения результатов вычислений, т. е. их содержимое может не приниматься во внимание при обращении к некоторой подпрограмме и по окончании ее использования. В таком случае обращение к подпрограмме может стоить так же дешево, как выполнение команды *перехода.* Сравните затраты, сопровождающие использование этого алгоритма, с затратами, связанными с реализацией нерекурсивных алгоритмов

Глава 9 АЛГОРИТМЫ ПРОРЕЖИВАНИЯ

9.1. ВВЕДЕНИЕ

Алгоритмы прореживания были и продолжают оставаться объектом пристального изучения в рамках проблематики обработки изображений и распознавания образов. Смысл применения алгоритма прореживания проиллюстрирован на рис. 9.1. Пикселы, помеченные крестиками, представляют результаты квантования

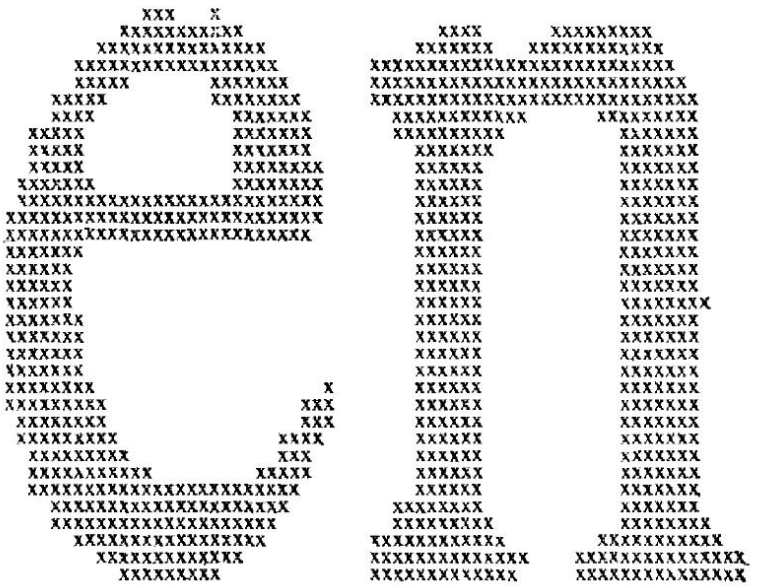


Рис. 9.1. Пример, иллюстрирующий необходимость использования алгоритмов прореживания

штрихового рисунка, который необходимо снова преобразовать в некоторое множество линий. Штриховой рисунок или текст должны быть подвергнуты квантованию с достаточно высоким разрешением с тем, чтобы можно было гарантировать отсутствие разрывов линий и сохранение концов последовательностей (см. гл. 7). Использование такого разрешения приводит к появлению в других частях изображения участков шириной более двух пикселов. Чтобы сохранить линейчатую структуру, свойственную исходному изображению, не нарушая его связности, необходимо использовать процедуру прореживания. В разд. 7.6 и 7.7 мы уже обсуждали, каким образом можно определить понятие «тонкость» на дискретной сетке. Этот анализ будет использован нами в качестве основы для определения алгоритмов прореживания.

На непрерывной плоскости тонкость можно математически строго определить следующим образом.

Определение 9.1. Пусть R — множество на плоскости, B — его граница и P — точка множества R. Ближайшим соседом точки P на границе B является такая точка M, принадлежащая границе B, что на этой границе нет никакой другой точки, расстояние от которой до точки было бы меньше расстояния PM. Если точка P имеет более одного ближайшего соседа, то P называют *остовной* точкой множества R. Объединение всех остальных точек называется *остовом*, или *серединной осью* множества R.

Из этого определения следует, что остовные точки являются центрами окружностей, полностью покрываемых множеством R, причем не существует окружностей с тем же центром и большим радиусом, покрываемых множеством R. На рис. 9.2 приведено

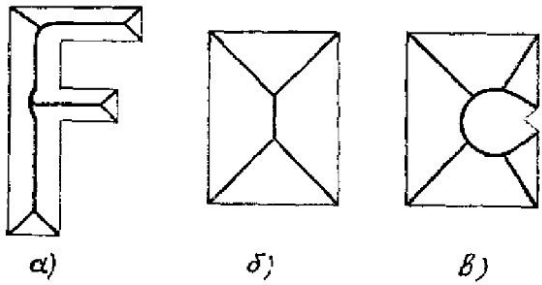


Рис. 9.2. Примеры остовов:

а — линейчатая структура силуэта точно соответствует структуре серединной оси; б — между структурой объекта и ребрами остова явного соответствия не имеется; в — наличие небольшого шумового воздействия приводит к резкому изменению формы остова

несколько примеров остовов, которые иллюстрируют и некоторые их основные свойства. Можно убедиться в том, что остовы чрезвычайно чувствительны к шуму, поскольку любое малое возмущение границы не только приводит к возмущению одного из ребер, но, кроме того, порождает новые ребра. Итак, имеет место следующий результат.

Утверждение 9.1. Если P — центр кривизны границы B плоского множества R в той точке границы, где ее кривизна имеет изолированный максимум, то соответствующий остов содержит ребро, оканчивающееся в точке P.

Доказательство. Наличие изолированного максимума кривизны означает, что радиус некоторой окружности, соприкасающейся с границей в этой точке (M), меньше, чем окружности, соприкасающейся с границей в соседних точках. Более того, по определению касания кривых, данная окружность имеет более одной общей точки с кривой и, следовательно, ее центр (P) входит в остов. Всякая окружность с большим радиусом будет иметь соприкосновение в двух точках, расположенных в окрестности точки M, и, следовательно, ее центр входит в остов, но будет отстоять от точки P.

Анализ рис. 9.2 позволяет прийти еще к одному выводу: если исходный объект является тонким (узким), то остов содержит су-

щественную информацию о его форме. Как показывает рис. 9.2,*б*, в случае толстых (широких) объектов это не так.

Перенос на дискретную плоскость понятия серединной оси не только не очевиден, но, быть может, и невозможен из-за осложнений, возникающих при определении равенства расстояний между пикселами на дискретной сетке. Следовательно, многое здесь зависит от интуиции разработчика алгоритма. Одна из возможностей заключается в обобщении определения 9.1 на дискретную плоскость. Можно определить какой-либо дискретный вариант окружности и отыскать «окружности», полностью покрываемые рассматриваемым множеством и обладающие тем свойством, что не существует больших «окружностей» с тем же центром, которые покрывались бы данным множеством. Поскольку реализация такого метода требует большого объема вычислений, он не получил широкого распространения. Большая часть литературы по прореживанию посвящена алгоритмам, определенным непосредственно на дискретной сетке. В данной главе мы рассмотрим два таких алгоритма. В основе обоих лежит понятие кратных пикселов (см. определение 7.9 и теорему 7.3), хотя в первом из них это определение используется лишь частично. При разработке обоих алгоритмов это понятие было несколько изменено таким образом, чтобы по мере возможности выделяемые остовы имели ширину одного пиксела. (Напомним, что встречаются линейчатые множества шириной в два пиксела — например, множество, представленное на рис. 7.17.) Поэтому при изложении сути этого подхода мы будем пользоваться в основном термином *«остовный пиксел»* вместо термина «кратный пиксел».

Определение 9.2. *Остовом* множества пикселов R называется множество, формируемое следующим образом. Сначала определяются пикселы остова и пикселы контура, принадлежащие множеству R. После этого все пикселы контура, не являющиеся остовными, удаляются и полученное в результате этой процедуры множество заменяет множество R. Этот процесс повторяется до тех пор, пока не будет сформировано множество, включающее только остовные пикселы.

Читатель, которого раздражает «кругообразность» этого определения, может вместо «остовный» поставить «кратный». Проверка пиксела на кратность требует изучения лишь его непосредственной окрестности, и, следовательно, преобразования, предусмотренные определением 9.2, могут быть реализованы с помощью локальных операций (см. подразд. 7.6.2). Существенным является допущение о том, что для каждого множества можно установить, какие пикселы принадлежат остову, и соответственно оставить их, а какие явно не принадлежат ему, и их соответственно можно исключить из дальнейшего рассмотрения. Напомним также, что это может быть определено без использования какой-либо заранее известной информации о принадлежности или непринадлежности пиксела контуру, поскольку все такие пикселы должны располагать n-соседом с нулевым значением метки.

9.2. КЛАССИЧЕСКИЕ АЛГОРИТМЫ ПРОРЕЖИВАНИЯ

Большинство алгоритмов, описанных в литературе, относят к категории остовных лишь такие пикселы, которые соответствуют конфигурациям, приведенным на рис. 7.20 и 9.3.*а*[1], т. е. пикселы, удовлетворяющие условию (а) теоремы 7.3. Для того чтобы не пропустить пикселы, удовлетворяющие другим условиям этой теоремы, в них используется очень специфический способ проверки условия (а). Отметим, что условие (а) представляет собой, в сущности, некоторый критерий связности и его последовательное применение к некоторому изображению приведет к стягиванию односвязной (т. е. не имеющей отверстий) области в один пиксел — очевидно, это не тот результат, к которому мы стремимся.

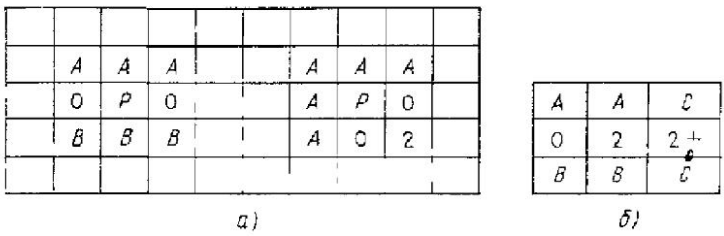


Рис. 9.3. Конфигурации окрестностей кратных пикселов

а - по меньшей мере один из каждой группы пикселов, помеченных символами *A и B* должен иметь ненулевое значение, *б* — по меньшей мере один из пикселов, помеченных символом *C*, должен иметь ненулевое значение Если оба пиксела, обозначенные символом *C*, имеют ненулевые значения, то пикселы, обозначенные символами Л и В, могут принимать любые значения В противном случае, по меньшей мере, один из пикселов каждой пары пикселов, обозначенных символами *A* или *B*, должен иметь ненулевое значение

Такое затруднение можно преодолеть, осматривая пикселы параллельно и обобщая определение 9.2 посредством добавления в него условия, согласно которому пиксел, отнесенный к категории остовных в процессе одной из итераций, не может быть после этого исключен из рассмотрения. Так, если Q — горизонтально расположеный массив пикселов, то удаление из его середины одного из пикселов приводит к нарушению условия связности. Следовательно, удаляться будут лишь два концевых пиксела массива, а остальные будут считаться элементами окончательного решения. Остается, однако, одна трудность: если массив состоит из пар пикселов типа приведенных на рис. 7.17, то ни один из них не является определяющим с точки зрения связности, и поэтому все они будут удалены. Последнее препятствие можно преодолеть, используя сочетание параллельной и последовательной обработок. Одновременной проверке подвергаются не все пикселы контура, *а* только те, N-соседи которых имеют нулевую метку, где N прини-

[1] На рис. 9.3 повторно воспроизводятся рис. 7.20 и 7 25, поскольку нам придется часто обращаться к ним в данной главе.

мает последовательно значения 0, 2, 4 и 6. Тогда массив, толщина которого равна двум пикселам, будет в первую очередь посредством прореживания превращен в массив толщиной в один пиксел, и, таким образом, часть пикселов этого массива будет сохранена. При использовании алгоритмов такого типа разметку остовных пикселов необходимо выполнять специальным образом с тем, чтобы предотвратить их удаление при какой-либо из очередных итераций вследствие их некритичности с точки зрения связности. Этот метод реализован в алгоритме 9.1. Он предназначен для работы с двухуровневым изображением, пикселы которого могут быть снабжены метками 0 или 1. Пикселы, составляющие изображения, могут в процессе прореживания получать также метки 2 или 3, так что при изучении конфигураций окрестностей такие значения меток следует считать индикатором наличия пиксела. Процедуру сопоставления конфигураций окрестностей, выполняемую на шаге 9, легко реализовать, если сформировать цепочку из меток восьми соседей пиксела p. В этом случае приходится сравнивать всего лишь две цепочки символов. Пример применения данного алгоритма приведен на рис. 9.4.



Рис. 9.4. Пример, характеризующий работу классического алгоритма прореживания: символ «—» — удаленные пикселы, символ «О» — пикселы остова

Алгоритм 9.1. Классический алгоритм прореживания

Обозначения. I — изображение, подаваемое на вход алгоритма. P — множество конфигураций окрестностей остовных пикселов, представленных на рис. 9.3,*а*, и конфигураций, полученных в результате поворота первой конфигурации на 90°

и трех последовательных поворотов второй на 90°. Истинное значение признака *оставление* используется для обозначения того, что пикселы, не принадлежащие остову, могут быть оставлены. Признак *ост* принимает истинное значение в случаях, когда окрестность пиксела соответствует одной из конфигураций окрестностей, входящих в множество *P*. Единица в описании конфигурации соответствует пикселу окрестности, имеющему ненулевое значение.

1. Присвоение признака *оставление* истинного значения.
2. **While** признак *оставление* имеет истинное значение **do** шаги 3—12.

Begin.

3. Присвоение признаку *оставление* ложного значения. {Никакие изменения не производились.}
4. For j = 0, 2, 4 и 6 do шаги 5—12.

Begin.

5. For всех пикселов p изображения / do шаги 6—10,

Begin.

6. If значение p равно 1 and **if** значение его j-соседа равно 0 **then do** шаги 7—10.

Begin.

7. Присвоение признаку *ост* ложного значения.
8. For всех конфигураций окрестностей, входящих в P **do** шаг 9.

Begin.

9. If конфигурация окрестности пиксела p соответствует одной из конфигураций P, **then** присвоение признаку *ост* истинного значения и выход из цикла.
   End.
10. If признак *ост* имеет истинное значение, **then** присвоение пикселу p значения 2 {остовный пиксел}, **else** присвоение пикселу p значения 3 {удаляемый пиксел} и, кроме того, присвоение признаку *оставление* истинного значения. **End. End.**

И. For всех пикселов p изображения / **do** шаг 12. -

Begin.

12. **If** значение пиксела p равно 3, then присвоение пикселу p значения 0.

End. End.

End.

13. **Конец алгоритма.**

9.3. АСИНХРОННЫЕ АЛГОРИТМЫ ПРОРЕЖИВАНИЯ

В классическом алгоритме прореживания операции обработки изображения выполняются в некоторой жесткой последовательности, однако это ограничение нетрудно устранить. Обработка изображений в чисто параллельном режиме не очень эффективна, поскольку большинство алгоритмов на самом деле работают вблизи границ и поэтому процессоры, предназначенные для обработки пикселов, расположенных внутри однородных участков изображения с большими размерами, большую часть времени будут бездействовать. С другой стороны, обрабатывая с помощью одного процессора в последовательном режиме матрицы, скажем, размерами 32X32 пиксела, можно добиться более равномерного распределения загрузки процессоров. Принятие подобного подхода налагает ряд ограничений на используемые в его рамках алгоритмы.

1. Поскольку предусматривается проведение параллельной об работки, нельзя пользоваться алгоритмами сугубо последователь ного характера.

2. Чтобы избежать возникновения проблем, связанных с обра боткой стыков зон, обрабатываемых двумя различными процес сорами, необходимо предусмотреть возможность осмотра каждым процессором части зоны, обрабатываемой соседним процессором, однако процессоры не должны производить какие бы то ни было изменения в чужих зонах. Следовательно, алгоритм должен иметь возможность изменять значение текущего обрабатываемого пик села, но не значения соседних пикселов. (Это требование суще ственно для всех параллельных алгоритмов, следует избегать си туации, когда два или более процессоров «пытаются» вводить ин формацию в одну и ту же ячейку памяти. Статус «текущего пик села» может, однако, меняться в зависимости от типа алгорит ма) Для того чтобы ограничить объем обмена между процессора ми, можно также ввести условие, предусматривающее осмотр про цессором только пикселов, расположенных вокруг текущего.

3. Поскольку обработка частично проводится последовательно, используемые данные и алгоритм должны быть организованы та ким образом, чтобы присвоение пиксела меток не оказывало влияния на решения, принимаемые на последующих шагах алго ритма, по крайней мере, до момента, когда процессоры могут быть вновь синхронизированы Хранение двух экземпляров матрицы, представляющей изображение (одного — для выборки данных, другого — для записи результатов обработки), является неэф фективным решением.

Любой алгоритм, удовлетворяющий трем перечисленным условиям, обладает и тем преимуществом, что его можно использовать как в чисто последовательном, так и в чисто параллельном режимах. Оказывается, условия теоремы 7.3, определяющие кратные пикселы, можно применять для синтеза алгоритмов данного типа.

Единственная проблема, возникающая в связи с реализацией данного подхода, заключается в том, что требуется редактирование получаемого остова, так как в конфигурациях шириной в два пиксела сохраняются оба пиксела. Толщину в один пиксел можно обеспечить, введя предпочтения по ориентации и удаляя по одному пикселу из каждой пары смежных кратных пикселов, если удаление соответствующего пиксела не отражается на связности остова.

Определение 9.3. Если 4- или 2-сосед кратного пиксела имеет нулевое значение метки, то последний квалифицируется как удаляемый пиксел при условии, что у него нет соседа, который был бы уже квалифицирован как удаляемый.

Последнее ограничение необходимо для предотвращения получения несвязных объектов в случаях, аналогичных представленному на рис. 9.5. Здесь внутренние пикселы обозначены единицей; пикселы контура, не соответствующие ни одной из существенных конфигураций, обозначены двойкой; пикселы контура, соответствующие одной из конфигураций, приведенных на рис. 9.3,*а*, обозначены тройкой; пикселы контура, соответствующие конфигурации на рис. 9.3,*б*, обозначены четверкой или символом *d*. Оба пиксела, снабженные меткой 4, имеют 4- или 2-соседа с нулевой меткой, однако один из этих пикселов следует оставить. **Алгоритм 9.2. Основной алгоритм прореживания**

Обозначения. R — множество пикселов исходного изображения, значения которых равны 1. $B(R)$ — граница множества R, $M(R)$ — множество кратных пикселов, входящих в множество R.

1. **Repeat** шаги 1—4.
Begin.
2. Определение множества $B(R)$, образуемого всеми пикселами, входящими в R, которые имеют н-соседа вне множества R
3. Определение множества $M(R)$, образуемого всеми кратными пикселами множества $B(R)$.
4. If $B(R)$ совпадает с $M(R)$ **then exit.**
5. Замена множества R множеством $R—(B(R)—M(R))$,
End.
6 **Конец алгоритма.**

В алгоритме 9.2 реализуется определение 9.2. Он не зависит от способа отыскания кратных пикселов.

1	1	2	0	0	0
2	2	*d*	0	0	0
0	0	4	4	3	3
2	2	1	*d*	0	0
1	1	1	2	0	0

Рис 9 5. Конфигурация, иллюстрирующая необходимость использования дополнительной метки при определении возможности удаления *кратного* пиксела

9.4. РЕАЛИЗАЦИЯ АСИНХРОННОГО АЛГОРИТМА ПРОРЕЖИВАНИЯ

Проверка выполнения трех условий, входящих в теорему 7.3, может выполняться следующим образом. Введем пять восьмибитовых регистров bdr_1, bdr_2, bdr_3, bdr_4 и bdr_5, значения каждого разряда которых задаются в соответствии со значениями соседей пиксела. 0-сосед ставится в соответствие самому старшему разряду, 1-сосед — следующему по старшинству разряду и т. д. (используемые обозначения соответствуют рис. 7.4). Если значение пиксела равно j, то задаются значения соответствующих битов во всех байтах bdr_k, $k<=j$. Первоначально все пикселы имеют значения 0 или 1, и поэтому можно задать значения лишь в байте bdr_1. Если выясняется, что пиксел принадлежит контуру, то его значение заменяется на двойку, в результате чего становится возможным задание значений в байте bdr_2. Конфигурации, приведенные на рис. 9.3, нетрудно представить как результаты побитового применения масок к указанным регистрам. Маски задаются в виде восьмеричных чисел. Например, первая из конфигураций рис. 9.3,*а* соответствует числу 160_8. Мы будем использовать запись *a&b* для обозначения побитовой операции *И*, примененной к цепочкам *a* и *b;* таким образом, в цепочке, полученной в результате операции $bdr_1\&160_8$, единичные значения разрядов будут соответствовать ненулевым значениям 1-, 2- и 3-соседей текущего пиксела. Для работы с конфигурациями, полученными в результате поворота исходной на 90°, необходимо просто осуществить в каждом регистре сдвиг на два разряда. Поэтому ниже мы не будем больше заниматься такими конфигурациями. (Каждому заданному критерию соответствует еще один или три критерия, полученные при помощи указанной операции сдвига.) Имеет место следующий простой результат.

Утверждение 9.2. Справедливы следующие обратные им утверждения:

(а) пиксел принадлежит контуру, если результат операции $bdr_1\&252_8$ не равен 252_8;

(б) окрестность пиксела соответствует левой конфигурации, приведенной на рис. 9.3,*а* если $bdr_1\&160_8<>0$, $bdr_1\&7<>0$ и $bdr_1\&\&210_8 = 0$;

(в) окрестность пиксела соответствует правой конфигурации, приведенной на рис. 9.3,*а* если $bdr_1\&202_8 — 0$ и $bdr_1\&1<>0$;

(г) пиксел не имеет соседней внутренней области, если запись в bdr_1 тождественна записи в bdr_2;

(д) пиксел соответствует требованиям конфигурации, приведенной на рис. 9.3,*б*, если $bdr_2\&200_8<>0$ и $bdr_1\&10_8 = 0$ и если выполняется одно из следующих условий.

(д1) результаты операций $bdr_1\&100_8$ и $bdr_1\&1$ больше 0; (д2) один из результатов операций, включенный в условие (д1), больше 0 и оба результата операций $bdr_1\&60_8$ и $bdr_1\&6$ больше 0.

Доказательство. Все перечисленные утверждения можно доказать при помощи непосредственного использования соответствующих определений. Например, в утверждении (д) цепочка 200а выделяет 0-соседа (самый старший бит маски является единичным). Наложение этой маски на bdr_2 с помощью операции *И(&)* дает 1, если 0-сосед принадлежит границе, и 0 — в остальных случаях. Цепочка 10_8 выделяет 4-соседа и наложение этой маски на bdr_1 с помощью операции *И* дает 1, если этот пиксел имеет ненулевое значение, и 0 — в остальных случаях. Следовательно, два эти условия служат критерием конфигурации рис. 9.3,*б*, образованной тремя пикселями, которые находятся в среднем ряду.

Эти результаты использованы в алгоритме 9.3 На шаге 4 отыскиваются пикселы, находящиеся на границе; на шаге 5 проверяется часть условия (б) теоремы 7.3, а на шаге 6 проверяется условие, содержащееся в утверждении 9.2 (б). На шаге 8 проводится разметка пикселов, не имеющих соседей внутри области, а на шаге 9 проводится проверка соответствия конфигурации, приведенной на рис. 9.3,*б*. На шагах 11 и 12 проверяются условия определения 9.3. На шаге 15 удаляются все пикселы со значениями меток 2 и 5. Шаги 6, 9, 11 и 12 предусмотрены для обработки конфигураций, имеющих другие ориентации (последние воспроизводятся посредством сдвига битовых комбинаций на два разряда).

Этот алгоритм без всяких затруднений можно реализовать, сочетая параллельную и последовательную обработки, поскольку процедура разметки не влияет на содержимое младших регистров, т.е. значение байта bdr_3 не зависит от значения байта bdr_k при $k>j$. Для обнаружения контура и проверки отдельных условий необходимо располагать лишь байтом bdr\. Следовательно, каждый процессор может выполнять ряд операций сопоставления цепочек символов в асинхронном режиме, а затем должен ждать, пока все процессоры не завершат этап обработки, прежде чем продолжать исполнение алгоритма.

Алгоритм 9.3. Асинхронный алгоритм прореживания

Обозначения. C — значение счетчика числа итераций, увеличенное на 5. Признак *удаление* указывает, подвергались ли пикселы удалению.

1. Присвоение счетчику *C* значения 5 и признаку *удаление* истинного значения.
2. While признак *стирание* имеет истинное значение do:
   Begin.
3. Увеличение значения счетчика C
4. Определение пикселов контура в соответствии с условием утверждения 9.2 (а) и для каждого из них do
   Begin.

5. If байт bdr_1 содержит не более одного ненулевого бита, **then** присвоение соответствующему пикселу метки 3.
If условия (б) или (в) утверждения 9.2 выполняются, then соответствующему пикселу присваивается метка 3.
6. **End.**
7. **Wait. For** всех пикселов **do:**
Begin.
8. If байт $\mathbf{bdr_1}$ тождествен байту $\mathbf{bdr_2}$ **then** присвоение соответствующему пикселу метки 4.
9. If некоторый пиксел принадлежит контуру и If условие (д) утверждения 9.2 выполняется **then** присвоение соответствующему пикселу метки 4.
End.
10. **Wait. For** всех пикселов **do:**
Begin.
11. **If** результат операции $bdr_4 \& 10_8$ равен 0, then присвоение соответствующему пикселу метки 5.
12. **If** результат операции $bdr_4 \& 40_8$ равен 0 и байт bdr_5 — нулевой, **then** присвоение текущему пикселу метки 5.
13. **If** пиксел имеет метку 3 или 4, **then** он классифицируется как остовный и ему присваивается метка, значение которой равно значению счетчика числа итераций C.
End.
14. **Wait. For** всех пикселов do:
Begin.
15. Удаление всех пикселов с метками 2 или 5. Если не удаляется ни один пиксел, признаку *удаление* присваивается значение «ложный».
End. End.
16. **Конец алгоритма.**

Данный вариант алгоритма предполагает, что каждому процессору предъявляется часть изображения, причем он может производить выборку значений пикселов, примыкающих (извне) к границам обрабатываемой им зоны, однако этот процессор не может записывать какую бы то ни было информацию в те участки памяти, где хранятся указанные пикселы. Следовательно, не требуется применения каких-либо специальных средств кадрирования, за исключением случаев выхода за пределы собственной рабочей зоны. Ключевое слово **wait** (ожидание) означает, что на соответствующих шагах каждый процессор ждет окончания работы остальных процессоров. При решении некоторых приклад-

ных задач может потребоваться сохранять информацию о толщине (ширине). В данном алгоритме это легко предусмотреть, использовав номер итерации в качестве значения метки каждого пиксела при его первом заключении в категорию основных. Для этого потребуются дополнительные затраты памяти, однако во всех остальных отношениях реализация этой модификации алгоритма оказывается очень простой. Символ C обозначает число итераций, увеличенное на пять, и поэтому в качестве меток для пикселов, принадлежащих остову, могут использоваться числа не пересекающиеся со значениями меток, необходимых для выполнения основных операций алгоритма. Подобную разметку можно использовать для восстановления приближения исходной области по ее остову (см. задачу 9.5). На рис. 9.6 приведен пример применения описанного алгоритма к дискретизированным изображениям буквенно-цифровых символов.



Рис. 9.6. Результат применения асинхронного алгоритма прореживания: символ «—» - удалённые пикселы, символ «О» — пикселы остова

9.5. БЫСТРЫЙ АЛГОРИТМ ПРОРЕЖИВАНИЯ

Алгоритмы, рассмотренные в предыдущих разделах, осуществляют процедуру прореживания, систематически обрабатывая все изображения целиком. Встречаются, однако, такие прикладные задачи, при решении которых хранение изображения в памяти оказывается непрактичным, а прореживание целесообразно проводить более простым способом. Алгоритм 9.4 реализует чрезвычайно простой способ прореживания, но при обработке многих объектов выделить части остова с его помощью не удается. С

другой стороны, он обладает быстродействием, существенно большим, чем любой из описанных в предыдущих разделах алгоритмов, и, возможно, было бы неплохо использовать такой алгоритм в качестве средства предварительной обработки, причем даже в тех случаях, когда объекты на изображениях не представлены, главным образом, тонкими линиями. В такой ситуации к частям силуэта, оставшимся непрореженными, можно применить один из универсальных алгоритмов прореживания.

Данный алгоритм выполняет последовательную построчную обработку, причем в памяти должно храниться лишь небольшое число строк или их частей. Для простоты будем описывать работу алгоритма как процесс обхода графа смежности строк, хотя запоминание последнего в явном виде не предусматривается. (Анализ параллельного режима работы проведен в гл. 8). Для данного алгоритма весьма существенным являются понятия вертикального, горизонтального и диагонального маршрутов, которые мы определим следующим образом.

Определение 9.4а. Вертикальным маршрутом ГСС называется такой маршрут, ширина всех вершин которого меньше значения заданного порога w_0 и число вершин которого больше заданного значения N_0.

Определение 9.4*б* Горизонтальным маршрутом ГСС называется такой маршрут, ширина всех вершин которого больше порогового значения $N_0 v_s$, число вершин которого меньше значения $w_0 l v_s$, а вершины, расположенные по обеим сторонам маршрута, имеют ширину, не меньшую порогового значения w_0 (v_s — расстояние между линиями просмотра).

Определение 9.4в. Диагональным маршрутом ГСС с углом наклон φ называется такой маршрут, все центры вершин которого приближенно коллннеарны (чтобы убедиться в этом, можно воспользоваться одним из алгоритмов, приведенных в гл. 12), угол наклона прямой, проходящей через эти центры, равен φ, ширина всех вершин меньше значения $w_0 \sin \varphi$ и число вершин больше значения $N_0 \cos \varphi$

Пример 9.1. На рис. 9.7,*а* представлен силуэт изображения и соответствующий ГСС. Видно, что два ребра точно выделены и идентифицированы алгоритмом 9.4. Поскольку ширина всех вершин мала, подмаршрут 1—7 можно считать вертикальным штрихом Вершина 8 сама по себе образует подмаршрут, который можно рассматривать как горизонтальный штрих Если бы вершина 7 не входила в первый подмаршрут, ее можно было бы использовать в качестве вершины, связывающей два выделенных штриха. Пример, приведенный на рис. 9.7,*б*, иллюстрирует случай, когда с помощью данного алгоритма не удалось выделить криволинейный штрих Вершину 3 как таковую можно было бы считать горизонтальным штрихом, однако, наличие вершины 4 исключают вынесение такого классификационного решения

Алгоритм 9.4. Быстрый алгоритм прореживания

1 Формируется ГСС и каждой его вершине присваивается метка, указывающая местоположение центра и ширину соответствующего интервала.

2. Для выделения всех маршрутов, образованных вершинами *с* порядками (1.1) и, возможно, начинающихся и/или оканчивающихся в вершинах с порядками (1,0) или (0,1), используется какой-либо алгоритм обхода графа (например, алгоритм 6.2).
3. While будет установлено, что любое из условий проверяемых на шагах 5, 6 и 8, выполняется, do
   Begin.
4. For каждого маршрута, выделенного на шаге 2, do:
   Begin.
   If маршрут содержит вертикальный подмаршрут, then все вершины данного подмаршрута следует снабдить меткой принадлежности к соответствующему ребру.
6. If маршрут содержит горизонтальный маршрут, then все вершины данного подмаршрута следует снабдить меткой принадлежности к соответствующему ребру.
7. If ни одно из условий, проверяемых на шагах 5 и 6, не выполняется, then do:
   Begin.
8. Проводится поиск диагонального подмаршрута. If таковой обнаруживается в угол его наклона равен φ, then все вершины этого подмаршрута должны быть снабжены соответствующими метками.
   End. End.
   End
9. **Конец алгоритма.**

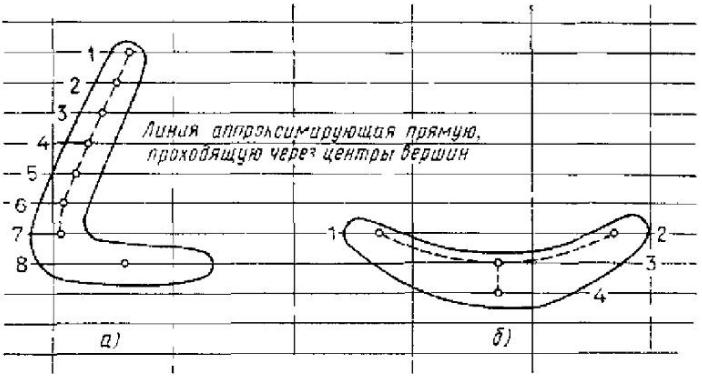


Рис. 9.7. Примеры силуэтов, применение к которым алгоритма 9 4 приводит к успеху **(а)** или заканчивается неудачей (б)

9.6. СТРУКТУРНЫЙ АНАЛИЗ ФОРМЫ ОБЪЕКТОВ ИЗОБРАЖЕНИЯ

В кратком введении в анализ формы объектов изображения, которому был посвящен разд. 7.8, мы отмечали, что во многих случаях восприятие формы объекта определяется его общей структурой. Результаты применения алгоритмов прореживания могут явиться основой соответствующего структурного анализа, и в данном разделе будет дан обзор возможностей такого метода. В общем случае в результате структурного анализа должны быть определены элементарные блоки, из которых строится объект, силуэт которого задается как часть изображения класса 2. Этот процесс можно выполнять непосредственно при помощи разбиения объекта на ряд непроизводных объектов простой формы. Например, силуэт можно представить как объединение его выпуклых подмножеств. Подобный путь, хотя теоретически и возможный, требует значительных затрат вычислительных ресурсов. Аналогичных результатов, но с меньшими затратами можно достичь, обратившись к анализу остовов объектов. Следует иметь в виду, что хотя с теоретической точки зрения данный метод можно применять к произвольному объекту, приемлемые результаты он позволяет получить лишь при работе с тонкими (узкими) объектами.

Анализ формы объектов, основанный на методах прореживания, может найти практическое применение при распознавании штриховых изображений или буквенно-цифровых символов, т. е. тех объектов, которые играют заметную роль в интерактивной машинной графике. С другими методами разбиения объектов на части можно познакомиться по литературе, упоминаемой в разд. 9.8. Основным препятствием для применения структурных методов является отсутствие подходящих алгоритмов классификации. Обсуждение этой проблемы в полном объеме выходит за пределы задач нашей книги и поэтому здесь мы лишь подытожим главные трудности, возникающие в этой связи.

1. Результаты структурного анализа лучике всего представлять в виде некоторого графа; их можно описывать, пользуясь графо выми грамматиками, однако этот способ оказывается не очень плодотворным, несмотря на его теоретическую привлекатель ность. Объясняется это, по меньшей мере, двумя причинами: во-первых, очень трудно выводить грамматику по выборочным дан ным и, во-вторых, при использовании графовых грамматик чрезвы чайные затруднения вызывает проведение синтаксического ана лиза.

2. Альтернативный применению графовых грамматик является задание на (помеченном) графе ряда логических функций, на пример «наличие двух вертикальных штрихов». После этого мож но сформировать дерево решений или определить двоичные век торы, компоненты которых соответствуют заданным логическим функциям. Эти векторы можно затем классифицировать, исполь-

зуя соответствующие статистические решающие правила. Проблемой в данном случае является выбор логических функций, поскольку отсутствуют процедуры, позволяющие делать это «автоматически».

9.7. ПРЕОБРАЗОВАНИЕ ДВУХУРОВНЕВЫХ ИЗОБРАЖЕНИЙ В ШТРИХОВЫЕ

Процедура прореживания в чистом виде не дает полного решения задачи преобразования изображений класса 2 в изображение класса 3. Остовы, получаемые с помощью алгоритмов 9.1 и 9.3, представляют собой все еще двухуровневые изображения, хотя и содержащие очень немного зачерненных пикселов. Такое изображение эквивалентно графу, вершинами которого служат зачерненные пикселы, а ребра связывают пикселы с их соседями. Этот граф можно следующим образом преобразовать в некоторый другой граф, не содержащий вершины второго порядка.

Обход графа двухуровневого изображения, содержащего его остов производится с помощью алгоритма 6.2 или аналогичного ему. Помимо стека 5 предусматривается буферная память T, в которую заносятся координаты всех пикселов, соответствующих вершинам второго порядка. Если порядок некоторой вершины отличается от двух, координаты соответствующей точки выводятся и проверяется наличие точек в буферной памяти. Если какие-либо точки в ней обнаруживаются, выводится их описание, а также координаты точки, соответствующей последней просмотренной вершины, порядок которой отличается от двух. Мы проиллюстрируем порядок выполнения этого процесса на примере.

На рис. 9.8 представлен остов изображения, приведенного на рис. 9.6. Для обхода соответствующего графа необходимо в каждой связной компоненте задать некоторую исходную вершину. Такие вершины можно найти в процессе просмотра двухуровневого изображения, содержащего остовные пикселы, осуществляемого сверху вниз слева направо. В общем случае эти вершины имеют второй порядок, что, в частности, имеет место и для обеих компонент изображения рис. 9.8. В процессе обхода графа информация может не выводиться до тех пор, пока не будет обнаружена некоторая вершина, порядок которой отличается от двух. Если такую вершину обнаружить не удается, то соответствующий граф представляет собой просто некоторую цепь и при выводе можно ограничиться этой информацией. Допустим, что в данном примере для первой связной компоненты в качестве начальной точки выбрана вершина d, а для второй — вершина a. В процессе обхода, начатого в вершине d, обнаруживается, что C является следующей вершиной порядка, отличного от двух. В таком случае выводится пара вершин d и C, a также описание расположенных между ними пикселов, хранящееся в буферной памяти T. Для представления этого описания существуют различные способы.

Юдин из них предусматривает выдачу маршрута, образованного этими пикселами и представленного в цепном коде. В данном случае это описание имеет вид $5^3 6 5 4 5^2 4^2 5 4^5 3 4^2 3^4 2 3^3 2^9 1^2$; показатели степени обозначают повторение символов. Другой способ предусматривает выдачу описаний прямых и дуг окружностей. В данном случае такое описание будет включать радиус

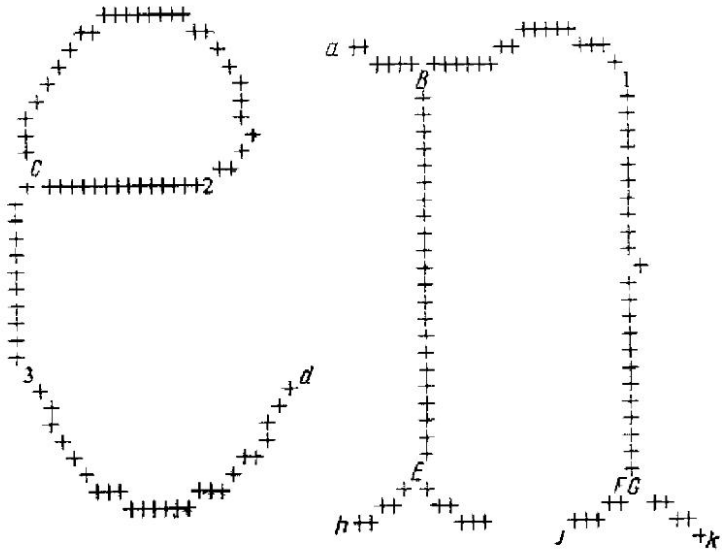


Рис. 9.8. Представление остова в виде некоторого графа вершины второго порядка обозначены символом «+» или числом, вершины первого порядка — строчными и вершины третьего порядка — прописными буквами

и центр некоторой дуги окружности с концевыми точками *d* и 3 и прямую, соединяющую точку 3 с точкой *C*. Для получения подобного описания приходится обращаться к методам вычерчивания кривой по точкам и, в частности, к методам кусочной аппроксимации, рассмотренным в гл. 12. В этом случае пиксел 2 выбирается с помощью алгоритма типа алгоритма 12.1.

Осмотр верхнего контура начинается с пиксела *C*, в котором он и заканчивается Соответствующее описание, представленное кривыми, включает радиус и центр дуги, соединяющей точки *C и 2*, и прямую проведенную между ними.

При обработке второй компоненты первое описание будет содержать пару пикселов *a* и *B* и соединяющее их ребро, представленное прямой. В процессе продвижения осмотра от точки *B* пер вой выделенной точкой является *E*, а затем — точка *h*. Для продолжения обхода необходимо извлечь какую-либо вершину из стека *S*, в данном случае таковой является правый нижний сосед вершины *E*. Далее следует ребро, соединяющее вершины *E* и *i* и т. д. Результаты такого обхода сведены в табл. 9.1. Последователь-

Таблица 91 Описание изображения, приведенного на рис. 9.8

Концевые точки	Описание ребра
d, 3	дуга, обращенная вогнутостью кверху
3, C	вертикальная прямая
C, 2	дуга, обращенная вогнутостью книзу
2, C	горизонтальная прямая
a, 8	диагональ, идущая слева направо
B, *E*	вертикальная прямая
E, *h*	диагональ, идущая справа налево
F, *i*	диагональ, идущая слева направо
B, 1	дуга, обращенная вогнутостью книзу
1, F	вертикальная прямая
F, *0*	«отрезок» слишком мал для формирования ребра[1]
F, *i*	диагональ, идущая справа налево
F, *k*	диагональ, идущая слева направо

[1] Точка G считается тождественной точке *F*.

ность появления ребер определяется порядком осмотра окрестности каждого пиксела. В данном случае просмотр велся в порядке возрастания номеров направлений, обозначенных на рис. 7.4. В описание ребер введена также информация, характеризующая их ориентацию. Для ее получения можно воспользоваться методами построения кривых по точкам, излагаемыми в следующих трех главах.

Мы не приводим в явном виде алгоритм построения указанных описаний, поскольку обход графа может осуществляться различными способами, выбор одного из которых определяется характером решаемой задачи.

9.8. БИБЛИОГРАФИЧЕСКИЕ ЗАМЕЧАНИЯ

Преобразование, приводящее к выделению серединных осей, впервые было описано в работе [9.3]. В статье [9.6] решение этой задачи на непрерывной плоскости дается посредством аппроксимации границ многоугольниками и последующего решения систем линейных уравнений, что позволяет определить годографы эквидистантных точек. Классический алгоритм прореживания представляет собой, в сущности, алгоритм, предложенный в статье [9.12] и затем усовершенствованный в работе [9.13]. Среди других первоначальных работ по этой тематике следует отметить статьи [9.4, 9.9—9.11]. Статья [9.14] посвящена изучению зависимости кратности обхода изображения от типа конфигурации окрестности. В статьях [9.1, 9.8] описаны последовательные алгоритмы прореживания, принцип действия которых состоит в удалении граничных пикселов. (Преимущество этих алгоритмов заключается в том, что они при реализации на универсальной ЭВМ

с жесткой последовательностью операций работают быстрее параллельных алгоритмов).

Примеры простых структурных описаний можно найти в работах [6.3, 9.2, 9.5, 9.7]. Использование ГСС в процедуре разбиения рассмотрено в статье [6.3]. Метод, изложенный в докладе [9.7], предусматривает, в сущности, построение остова на основе анализа ГСС. В статьях [9.2] описывается алгоритм непосредственного прореживания и его применение для распознавания рукописных цифр. Интересно отметить, что структурные описания находят довольно ограниченное применение при решении прикладных задач, хотя предложены они достаточно давно (еще в 1959 г.). Скорее всего это объясняется тем, что их использование сопровождается затратами вычислительных ресурсов, большими чем необходимо для применения чисто эвристических методов. Снижение стоимости обработки информации на ЭВМ должно, однако, привести к снижению стоимости этих методов. Следовательно, при разработке любой системы анализа изображений эти методы следует рассматривать в качестве потенциальных кандидатов на реализацию в системе. Более подробную информацию о структурных методах можно найти в монографиях [3.7, 7.2] и в сборнике [7.3].

9.9. ЗАДАЧИ

9.1. Напишите программу, реализующую алгоритм 9.1. 9

2. Дайте полное доказательство утверждения 9 2.

9.3. Можно ли уменьшить число меток, используемых в алгоритме 9.3? Можно ли, в частности, отказаться от использования для остовных пикселов метки 3?

9.4. Напишите программу, реализующую алгоритм 9.3.

9 5. Предложите алгоритм для восстановления толстых (широких) областей по остову, полученному с помощью алгоритма 9 3. Воспользуйтесь следующим правилом начиная с наибольших значений меток, все нулевые соседи всех пикселов с заданным значением метки заменяются пикселами со значением метки, на единицу меньшим заданного, после чего исходное значение метки заменяется на 1. Это правило последовательно применяется до тех пор, пока все ненулевые пикселы не будут снабжены единичной меткой. Сопоставьте результаты, полученные при разметке лишь непосредственных соседей, с результатами, получаемыми при разметке всех восьми соседей. Следующий пример иллюстрирует последовательность преобразований простой фигуры, осуществленных в соответствии с правилом непосредственного соседа (рис. 9.9).

9 6. Если Вы знаете ассемблер, напишите на нем программу для определения цепочек bdr_k и проверки битовых комбинаций, используемых в утверждении 9.2 и алгоритме 9 3. Предусмотрите операцию сдвига для того, чтобы была возможна проверка конфигураций, повернутых на 90°.

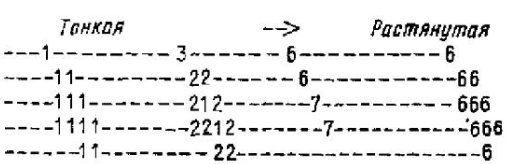


Рис 9.9 Пример к задаче 9.5. (Напоминаем, что алгоритм 9 3 прибавляет к порядковому номеру итерации число 5)

ПОСТРОЕНИЕ ПО ТОЧКАМ И ВОСПРОИЗВЕДЕНИЕ КРИВЫХ

10.1. ВВЕДЕНИЕ

В разд. 7.6. обсуждались возможности задания кривой на дискретной сетке, а в разд. 9.7 — способы построения кривой по ее остову. При решении одних прикладных задач допустимо представление кривой в виде последовательности пикселов, а при решении других предпочтительно задавать кривую с помощью математического выражения. Последнее может оказаться значительно компактнее дискретных представлений, как свидетельствует сопоставление рис. 9.8 и табл. 9.1. Отыскание кривой, проходящей через заданное множество точек, составляет задачу *интерполирования,* а отыскание кривой, проходящей вблизи заданного множества точек — задачу *аппроксимации.* Для обозначения обеих задач будем пользоваться термином *построение кривой по точкам.* Представляет также интерес задача *воспроизведения изображения кривой,* заключающаяся в следующем: для заданного математического выражения требуется найти соответствующие пикселы и присвоить им значения, при которых они составляют изображение кривой, описываемой заданным выражением. Эту задачу нельзя считать тривиальной даже при воспроизведении изображений прямых (см. разд. 7.6).

Эти задачи часто встречаются в какой-либо комбинации. Так, множество точек, по которым должна строиться кривая, может быть не плотным, как в примере, приведенном на рис. 9.9, а разреженным. При этом, соединяя заданные точки кривой и отображая их на сетке, приходится затрачивать на эту кривую много больше пикселов, чем имеется точек в исходном множестве.

Задачи построения кривых по точкам возникают в проектировании и промышленном производстве, а также в машинной графике, обработке изображений и распознавании образов. Например, очертания кузова автомобиля могут быть заданы с помощью множества дискретных точек, выбранного на основе технических и эстетических соображений. Для того чтобы ЭВМ могла управлять обрабатывающими инструментами, необходимо располагать математическим описанием гладкой поверхности, проходящей через все заданные точки. Этот пример отражает одно из первых применений методов машинной графики в промышленности. Другие возможности применения этих методов связаны с представлением экспериментальных данных для их последующего воспроизведения или автоматического распознавания. В последнем случае математическое описание контура объекта может содержать информацию о классе, к которому объект принадлежит. Точные требования, которым должны удовлетворять воспроизводимые

кривые и поверхности, зависят от конкретной прикладной задачи, однако в целом решение подобных задач базируется на общей методологии, которая и явится предметом нашего рассмотрения в данной и трех следующих главах.

С математической точки зрения задачи интерполирования, вероятно, решать легче, однако при решении многих прикладных задач аппроксимация оказывается более практичной, так как точные значения обрабатываемых данных искажаются из-за наличия шума. Компромиссным решением при выборе одного из этих методов служит выделение множества точек-ориентиров, которые могут быть определены пользователем в интерактивном режиме, и проведение кривой (или поверхности) вблизи этих точек. Ниже мы уделим внимание этому подходу. Удовлетворительное воспроизведение кривых требует решения ряда трудных задач из дискретной геометрии, однако обычно используются *частные* решения, дающие разнообразные качественные результаты.

Часто решающее значение при построении кривых по точкам приобретает выбор математического описания (функции). Хотя многочлены — первое, что приходит здесь в голову, их применение обычно дает плохое решение. Наибольшее распространение получили методы, предусматривающие использование кусочно-полиномиальных функций различных типов. При решении задач аппроксимации также следует уделять внимание выбору критерия, характеризующего качество приближения. Максимальное расстояние точек от кривой или поверхности представляется вполне разумным критерием, однако часто его использование порождает сложные вычислительные проблемы. В принципе, необходимо достижение некоторого компромисса между тем, что интуитивно кажется желательным, и тем, что оказывается реальным с вычислительной точки зрения.

Начнем с построения кривых по точкам при помощи многочленов. В двух следующих главах будет рассмотрено построение кривых по точкам кусочно-полиномиальными методами. Основное внимание будет сосредоточено на аппроксимации B-сплайнами и многоугольниками, а в данной главе мы займемся многочленами Безье. Некоторые методы построения поверхностей рассматриваются в гл. 13.

Для эффективного воспроизведения кривых необходимо, чтобы точки, составляющие отображение, порождались на одном из низших уровней, как правило, с помощью аппаратной части дисплея. Это обстоятельство ограничивает число классов кривых, поддающихся эффективному воспроизведению. Чаще мы имеем дело с кривыми двух классов: прямыми линиями и дугами окружностей. (В случаях, когда путаница исключена, будем называть их линиями и дугами соответственно). Для большинства прикладных задач достаточно этих двух типов кривых, поскольку на их основе можно воспроизводить и более сложные кривые. Задачи, связанные с воспроизведением кривых, будут рассмотрены в разд. 10.7.

10.2. ИНТЕРПОЛИРОВАНИЕ С ПОМОЩЬЮ МНОГОЧЛЕНОВ

Пусть (x_1, y_1), (x_2, y_2), ..., (x_n, y_n) — последовательность точек, заданных на плоскости, причем $x_i <> x_j$ при $i <> j$. Для таких точек можно непосредственно написать формулу интерполяционного многочлена $(n-1)$-й степени:

$$p_n(x) = y_1 \frac{(x-x_2)\ldots(x-x_n)}{(x_1-x_2)\ldots(x_1-x_n)} + y_2 \frac{(x-x_1)(x-x_3)\ldots(x-x_n)}{(x_2-x_1)(x_2-x_3)\ldots(x_2-x_n)} + \\ + \ldots + y_n \frac{(x-x_1)\ldots(x-x_{n-1})}{(x_n-x_1)\ldots(x_n-x_{n-1})}; \qquad (10.1)$$

интерполяционный многочлен можно представить в более строгом виде:

$$p_n(x) = \sum_{i=1}^{n} y_i \prod_{j \neq i} \frac{(x-x_j)}{(x_i-x_j)}. \qquad (10.1)$$

Из приведенного выражения следует, что значение y_i умножается на дробь, равную 1 при $x = x_i$ и 0 при остальных значениях x, принимаемых им в заданных координатах. Частному случаю $n = 2$ соответствует уравнение линии, соединяющей две точки:

$$p_2(x) = y_1 \frac{x-x_2}{x_1-x_2} + y_2 \frac{x-x_1}{x_2-x_1}. \qquad (10.2)$$

Основным недостатком интерполирования с помощью многочленов являются существенные колебания, которые может претерпевать кривая, построенная между двумя точками. Это обстоятельство иллюстрируется следующим примером.

Пример 10.1. Заданы пять точек: (0,0), (1,3), (2,0), (3,0) и (4,0). Соответствующий интерполяционный многочлен имеет вид

$$p(x) = -\frac{1}{2} x(x-2)(x-3)(x-4) \qquad (10.3)$$

и представлен на рис. 10.1,*а*. Он имеет три экстремума, расположенных вблизи точек (0,67; 3,46), (2,46; —0,47) и (3,5; 0,66). Может потребоваться, чтобы

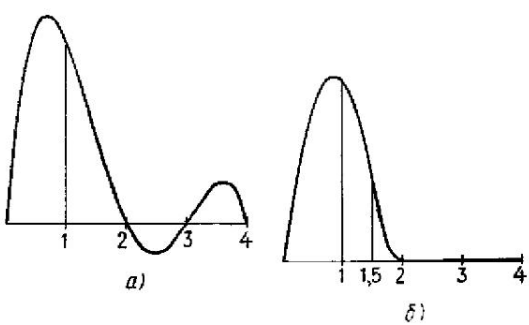


Рис. 10.1. Сравнение результатов интерполирования с помощью многочленов (а) и кусочно-многочленного интерполирования (б)

многочлен принимал близкие нулю значения на сегменте [2,4] и имел один из максимумов в точке $x=1$ или был симметричен относительно нее; однако данный многочлен ни одним из этих свойств не обладает.

Причина подобного поведения интерполяционных многочленов заключается в том, что многочлен, в принципе, представляет собой сумму степенных функций x. Эти функции обладают тем свойством, что их значения на всем сегменте определяются значением на произвольном малом подсегменте. Подбирая коэффициенты многочлена таким образом, чтобы сумма составляющих его функций принимала в нескольких точках искомые значения, мы не в состоянии контролировать значения отдельных членов в остальных точках. Поскольку значение каждого члена может быть достаточно велико, то появление значительных отклонений не должно являться неожиданным. Так, в примере 10.1 не удается обеспечить нулевое значение интерполяционного многочлена на сегменте [2.4]. Теперь, очевидно, целесообразно обратиться к интерполированию с помощью кусочных многочленов.

Пример 10.2. Для интерполирования тех же, что и выше, пяти точек используется кусочно-квадратичный многочлен. Вводится промежуточная точка (1,5; 1,35), и в результате решение выглядит следующим образом:

$$p_a(x) = 6x(0{,}6 - 0{,}7x), \qquad 0 \leqslant x \leqslant 1{,}5; \tag{10.4а}$$

$$p_b(x) = 5{,}4(x-2)^2, \qquad 1{,}5 \leqslant x \leqslant 2; \tag{10.4б}$$

$$p_c(x) = 0, \qquad 2 \leqslant x \leqslant 4. \tag{10.4в}$$

Методы выбора промежуточных точек обсуждаются в разд. 12.4. Здесь лишь отметим, что полученное таким образом приближение является непрерывным и имеет непрерывную первую производную — соответствующая гладкая кривая представлена на рис. 10.1,*б*. Хотя построенная функция не симметрична относительно $x=1$, она симметрична относительно $x=0{,}857$ и ее значение в этой точке равно 3,09. (Строгой симметричности относительно $x=1$ можно было бы добиться, выделив четыре сегмента: [0; 0,5], [0,5; 1,5] и т. д.)

Эти примеры показывают, что интерполирование с помощью многочленов имеет смысл только при сравнительно небольших сегментах. Оно заслуживает внимания лишь в том отношении, что образует в некотором смысле основу для более приемлемых способов построения кривых по точкам. Однако не следует считать, что кусочно-многочленное интерполирование всегда дает результаты лучшие, чем интерполирование с помощью многочленов. Если область определения данных делится неудачно, то все преимущества кусочной аппроксимации полностью исчезают.

Иногда помимо множества точек задается и касательная, которую должна иметь кривая, проходящая через эти точки, в каждой из них. Выражение, представляющее интерполяционный многочлен для этого случая, является весьма громоздким и мы не приводим его здесь. Читатель может найти его в учебниках по дополнительным главам математического анализа или по вычислительной математике. Мы ограничимся лишь выражением для случая, когда заданы две точки и касательные в них (x_1, y_1, y'_1) и (x_2, y_2, y'_2):

$$p(x) = \left(y_1' - \frac{y_1 - y_2}{x_1 - x_2} \right) \frac{(x - x_2)^2 (x - x_1)}{(x_1 - x_2)^2} +$$
$$+ \left(y_2 - \frac{y_1 - y_2}{x_1 - x_2} \right) \frac{(x - x_1)^2 (x - x_2)}{(x_1 - x_2)^2} + y_1 \frac{x - x_2}{x_1 - x_2} + y_2 \frac{x - x_1}{x_2 - x_1}. \quad (10.5)$$

В общем случае роль интерполяционного многочлена такою типа выполняет кубический многочлен. Он будет линейным в том и только том случае, если y'_1 и y'_2 равны угловому коэффициенту линии, соединяющей точки (x_1, y_1) и (x_2, y_2), и квадратичным в том и только том случае, если среднее от y'_1 и y'_2 равно тому же угловому коэффициенту.

Пример 10.3. Заданы точки (0,0) и (1,1) с угловыми коэффициентами О и s соответственно. Тогда

$$p(x) = (s - 2) x^3 + (3 - s) x^2, \quad (10.6)$$

На рис. 10.2 проиллюстрированы случаи: s=1 и s=16. Очевидно, что второй результат интерполирования неприемлем. С другой стороны, разумный результат для s=16 обеспечивается следующей кусочной аппроксимацией:

$$p_a(x) = 0, \qquad 0 \leqslant x \leqslant 7/8; \quad (10.7a)$$
$$p_b(x) = 64 (x - 7/8)^2, \quad 7/8 \leqslant x \leqslant 1. \quad (10.7b)$$

Соответствующий результат приведен на рис 10 2 (штриховая линия)

Данный пример показывает, что задание угловых коэффициентов может привести к очень значительным колебаниям, поскольку у многочлена производная некоторого порядка — константа (в данном случае — это третья производная, так как интерполяционный многочлен — кубический) и, следовательно, соответствующая функция не может очень резко изменяться.

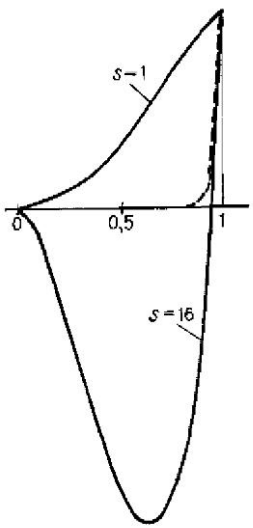


Рис. 102. Результаты интерполирования с помощью многочлена и кусочно-многочленного интерполирования при задании угловых коэффициентов в концевых точках кривой (сплошные линии); результаты кусочной аппроксимации для случая s=16 (штриховая линия)

10.3. МНОГОЧЛЕНЫ БЕЗЬЕ

Этот класс многочленов применяется в интерактивных системах машинной графики для приближенного решения задач на построение кривых по точкам. Вместо непосредственного использования точек, представляющих обрабатываемые данные, для задания многочлена при построении искомой кривой в интерактивном режиме определяется множество точек-ориентиров. Многочлены

при этом задаются не в явном виде как $y=p(x)$, а в параметрической форме:

$$x = p_x(t), \tag{10.8а}$$

$$y = p_y(t). \tag{10.8б}$$

Если (x_0, y_0), (x_1, y_1), ..., (x_m, y_m) — указанные точки-ориентиры, то соответствующий многочлен Безье определяется как

$$p_x(t) = \sum_{i=0}^{m} C_m^i t^i (1-t)^{m-i} x_i, \tag{10.9а}$$

$$p_y(t) = \sum_{i=0}^{m} C_m^i t^i (1-t)^{m-i} y_i, \tag{10.9б}$$

где C^i_m обозначает число сочетаний из m объектов по i:

$$C_m^i = \frac{m!}{i!\,(m-i)!}. \tag{10.10}$$

Эта формула неудобна для проведения вычислений — значение C^i_m лучше определять, пользуясь рекуррентной формулой

$$C_m^i = C_{m-1}^i + C_{m-1}^{i-1}. \tag{10.10'}$$

Уравнения (10.9) часто удобно записывать в векторной форме:

$$\mathbf{P}(t) = \begin{bmatrix} p_x(t) \\ p_y(t) \end{bmatrix}, \quad \mathbf{P}_i = \begin{bmatrix} x_i \\ y_i \end{bmatrix}, \tag{10.11}$$

таким образом,

$$\mathbf{P}(t) = \sum_{i=0}^{m} C_m^i t^i (1-t)^{m-i} \mathbf{P}_i. \tag{10.12}$$

На рис. 10.3 приведен один многочлен Безье для $m = 2$ и два многочлена Безье для $m=3$. Последний пример показывает, что размещение точек-ориентиров, обеспечивающее построение кривой определенного вида, требует достаточно высокой квалификации. Это обстоятельство является серьезным недостатком данного метода. Создается впечатление, что популярность приближе-

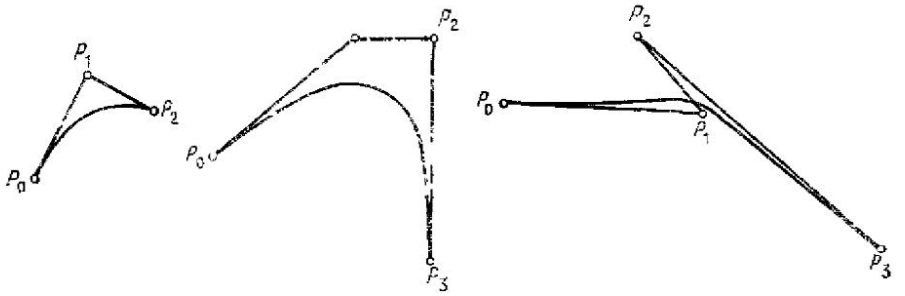


Рис. 10.3. Примеры многочленов Безье, построенных по трем и четырем точкам-ориентирам

ния с помощью многочленов Безье объясняется существенно большей простотой написания соответствующих программ ЭВМ по сравнению с программами, реализующими более совершенные методы построения кривых по точкам.

Уравнение (10 12) справедливо не только для двухмерных векторов P, определенных в соответствии с уравнением (10.11), но в той же степени для векторов с произвольной размерностью. Это удобно использовать при описании пространственных кривых. Из уравнения (10 12) следует, что

$$\mathbf{P}(0) = \mathbf{P}_0, \quad \mathbf{P}(1) = \mathbf{P}_m, \tag{10 13}$$

т. е. используется диапазон изменения t от 0 до 1.

Производная многочлена Безье, записанного в векторной форме равна

$$\mathbf{P}'(t) = -m(1-t)^{m-1}\mathbf{P}_0 + \sum_{i=1}^{\cdots} C_m^i [it^{i-1}(1-t)^{m-i} - (m-i)t^i(1-t)^{m-i-1}]\mathbf{P}_i + mt^{m-1}\mathbf{P}_m. \tag{10.14}$$

Отметим, что

$$\mathbf{P}'(0) = m(\mathbf{P}_1 - \mathbf{P}_0); \quad \mathbf{P}'(1) = m(\mathbf{P}_m - \mathbf{P}_{m-1}), \tag{10 14'}$$

и, следовательно, разложение в ряд Тейлора в окрестности нуля приводит к следующему выражению.

$$\mathbf{P}(t) = \mathbf{P}(0) + t\mathbf{P}'(0) + 0(t^2) \approx \mathbf{P}_0(1-mt) + mt\mathbf{P}_1; \tag{10 15a}$$

разложение в ряд Тейлора в окрестности единицы приводит к следующему выражению:

$$\mathbf{P}(t) = \mathbf{P}(1) - (1-t)\mathbf{P}'(1) + O(1-t)^2 \approx \mathbf{P}_m[1-m(1-t)] + m(1-t)\mathbf{P}_{m-1}. \tag{10 15б}$$

Итак, при $t \to 0$ и $t \to 1$ соответствующий многочлен Безье совпадает с линиями, соединяющими P_0 с P_1 и P_{m-1} с P_m, т. е. с касательными к данной кривой в точках P_0 и P_m. Более того, поскольку

$$\sum_{i=0}^{m} C_m^i t^i (1-t)^{m-i} = (t + 1 - t)^m = 1,$$

то значит многочлен Безье расположен внутри выпуклой оболочки множества точек-ориентиров.

10.4. ОПРЕДЕЛЕНИЕ МНОГОЧЛЕНА БЕЗЬЕ

Теперь выведем рекуррентное соотношение для многочленов Безье, имеющее наглядную графическую интерпретацию. Перепишем уравнение (10.12) в следующем виде.

$$\mathbf{P}(t) = (1-t)^m\mathbf{P}_0 + \sum_{i=1}^{m-1} C_m^i t^i (1-t)^{m-i}\mathbf{P}_i + t^m\mathbf{P}_m, \tag{10.16}$$

воспользуемся уравнением (10.10а) для разбиения суммы в уравнении (10.16) на две и получим

$$\mathbf{P}(t) = (1-t)^m \mathbf{P}_0 + \sum_{i=1}^{m-1} C_{m-1}^i t^i (1-t)^{m-i} \mathbf{P}_i +$$

$$+ \sum_{i=1}^{m-1} C_{m-1}^{i-1} t^i (1-t)^{m-i} \mathbf{P}_i + t^m \mathbf{P}_m$$

или

$$\mathbf{P}(t) = (1-t) \left[(1-t)^{m-1} \mathbf{P}_0 + \sum_{i=1}^{m-i} C_{m-1}^i t^i (1-t)^{m-1-i} \mathbf{P}_i \right] +$$

$$+ t \left[\sum_{i=1}^{m-1} C_{m-1}^{i-1} t^{i-1} (1-t)^{m-i} \mathbf{P}_i + t^{m-1} \mathbf{P}_m \right]. \qquad (10.17)$$

Анализ уравнения (10.17) показывает, что члены, заключенные в первой паре квадратных скобок, представляют собой многочлены Безье для точек P_0, P_1, ..., P_{m-1}, а члены, заключенные во второй паре квадратных скобок, — многочлен Безье для точек P_1, P_2, ..., P_m. Если обозначить многочлен Безье для точек P_k, P_{k+1}, ... P_l через $P_{kl}(t)$, то уравнение (10.17) можно переписать так:

$$\mathbf{P}_{0\,m}(t) = (1-t)\,\mathbf{P}_{0,\,m-1}(t) + t\,\mathbf{P}_{1m}(t) \qquad (10.18а)$$

или

$$\mathbf{P}_{0\,m}(t) = \mathbf{P}_{0,\,m-1}(t) + t\,[\mathbf{P}_{1m}(t) - \mathbf{P}_{0,\,m-1}(t)]. \qquad (10.18б)$$

Другими словами, многочлен Безье можно сформировать на основе двух других многочленов этого типа, соединив точки, соответствующие одному и тому же значению t, линией и разделив последнюю пропорционально значению t. Поскольку эту процедуру можно использовать для получения двух первых многочленов Безье, представим ее в виде специального алгоритма.

Алгоритм 10.1. Геометрический алгоритм построения многочленов Безье

Обозначения. На каждой итерации старые точки-ориентиры обозначаются через R_i, а новые — через Q_i. 1. For каждого t от 0 до 1 с интервалом At **do: Begin.**

2. For i от О до т задается $R_i = P_i$.
3. Задается $n = m$.
4. **While** $n>0$ **do:**
   Begin.
5. For i от 0 до n—1 задается $Q_i = R_i + t(R_{i+1} - R_i)$.
6. Значение n увеличивается на 1.

7. For $i = 0$ до n задается $Ri = Q_i$.
End.

8. $P(t)=R_0$.
End.

9. Конец алгоритма.

Этот алгоритм можно интерпретировать и графически, поскольку операция, выполняемая на шаге 5, эквивалентна выбору точки на сегменте, соединяющем точки R_i, и R_{i+1}, причем ее расстояние от точки R_i равно произведению текущего значения t на длину данного сегмента. Так, если $t = m / n$, то соответствующий сегмент делится на n частей и новая точка-ориентир размещается на конце такой m-й части, по порядку отстоящей от старой точки-ориентира. На рис. 10.4 иллюстрируется геометрический метод

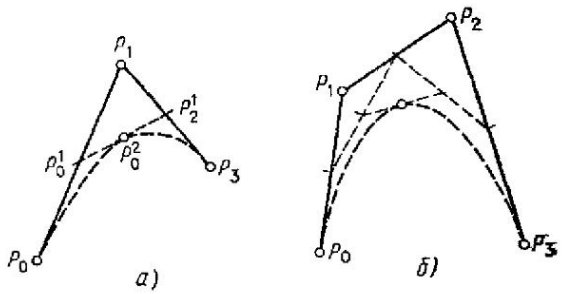


Рис. 10.4. Геометрический метод построения многочленов Безье при $t = 0{,}5$

Каждый новый набор точек-ориентиров определяется как множество точек, расположенных в серединах сторон многоугольника, построенного на предыдущем множестве таких точек Если остается лишь одна точка-ориентир, то это означает, что найдена точка искомой кривой

построения многочлена Безье для $m = 3$ и $m = 4$ при $t = 0{,}5$. Поскольку все векторы — двумерные, для выполнения шага 5 требуется затратить $6n$ скалярных операций: по одному сложению. вычитанию и умножению на компоненту вектора. В таком случае для выполнения всего алгоритма требуется

$$\sum_{n=1}^{m} (6n) - 3m\,(m+1) \qquad (10.19)$$

операций на каждое значение t, причем не требуется ни предварительной обработки, ни запоминания промежуточных результатов. При использовании уравнения (10.12) в сочетании с хранением таблиц значений t^i и $(1-t)^{m-i}$ и наличием значений биномиальных коэффициентов требуется затратить $m(m + 5)$ скалярных операций.

Еще более быстрый алгоритм можно получить, переписав уравнение (10.12) в следующем виде:

$$\mathbf{P}(t) = (1-t)^m \sum_{i=0}^{m} C_m^i \left(\frac{t}{1-t}\right)^i \mathbf{P}_i \qquad (10.20а)$$

или

$$\mathbf{P}(t) = (1-t)^m \left\{ \frac{t}{1-t} \left[\sum_{i=1}^{m} C_m^i \left(\frac{t}{1-t}\right)^{i-1} \mathbf{P}_i \right] + \mathbf{P}_0 \right\}. \qquad (10.20б)$$

Алгоритм 10.2. Построение многочленов Безье по схеме Горнера

1. **For** каждого t от 0 до 0,5 с шагом • t do:
   Begin.
2. Вычисляется $(1—t)^m$.
3. $Q_o = Pm$.
4. **For** i— 1 до «г **do:**
   Begin.
5. $$Q_i = t/(t-1) * Q_{i-1} + C^{m-1}{}_m P_{m-i}$$
   End.
6. $P(t) = (1—t)^m Q_m$.
   End.
7. For каждого t от 0,5 до 1 с шагом At **do:**
   Begin.
8. Вычисляется t^m.
9. $Q_0 = P_0$
10. **For** i**= 1 до** m **do:**
    Begin.
11. $$Q_i = t/(t-1) * Q_{i-1} + C^{m-1}{}_m P_{m-i}$$
    End.
12. $$P(t) = f^m Q^m.$$
    End. 13.

Конец алгоритма.

Для определения P(t) по значению члена уравнения (10.20*б*), заключенного в квадратные скобки, при условии, что известны значения $(1—t)^m$, $t/(1—t)$ и биномиальных коэффициентов, требуются восемь скалярных операций. Столько же операций необходимо затратить на вычисление каждого многочлена, заключенного в квадратные скобки и, следовательно, общее число затрачиваемых операций равно 8m+m=9m, причем второй член суммы характеризует число операций, затрачиваемых на вычисление $(1—t)^m$. Эта процедура, в сущности, представтяет собой применение схемы Горнера для вычисления многочленов Безье. При малых m(m<5) удобнее пользоваться геометрическим алгоритмом, поскольку его реализация требует небольшого числа вспомогательных операций, однако при больших m предпочтительнее схема Горнера. Чтобы увеличить точность решения, лучше использовать уравнение, действенное уравнению (10.20), вычисляя многочлен Безье при t>0,5. В этом случае множитель $t/(1—t)$ будет всегда меньше 1. Описанная процедура представлена в виде алгоритма 10.2.

Если помещать в одном месте более одной точки-ориентира, то можно увеличить точность приближения множества выбранных точек-ориентиров. Влияние этого приема на многочлен, представленный на рис. 10.4,*а*, иллюстрирует рис. 10.5. Очевидно, что использование кратных промежуточных точек оказывает глав-

ным образом локальное воздействие на результат. Для того чтобы подтвердить наличие такой тенденции аналитически, можно воспользоваться уравнением (10.14). Член, стоящий в квадратных скобках, представляет собой производную по t коэффициента при Pi в уравнении (10.12). Несложный подсчет показывает, что

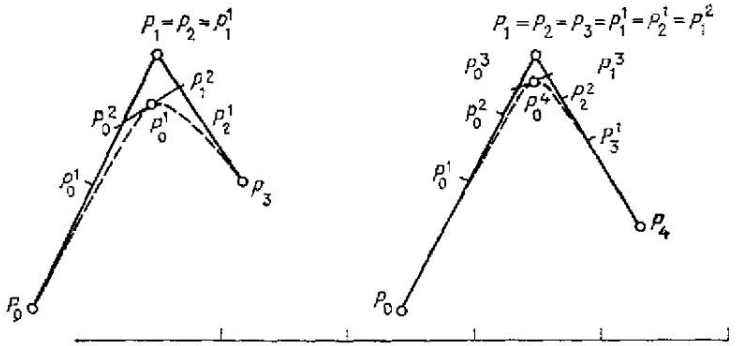


Рис 10.5. Иллюстрация геометрического метода построения многочленов **Безье** при использовании кратных точек-ориентиров

эта производная равна нулю при $t = t_i = i/m$ и, следовательно, коэффициент при P_i достигает максимального значения при $t—t_i$. Можно также убедиться в том, что значение этого коэффициента быстро уменьшается до 0 при t, отличных от t_i. Итак, для любого значения t имеется лишь несколько точек-ориентиров, которые влияют на форму искомой кривой: это точки, для которых t_i близко заданному значению t. Увеличение кратности точки ведет к расширению диапазона значений t, в котором данная точка является определяющей для формирования искомой кривой.

Содержательно многочлен Безье можно представить как некоторую намагниченную эластичную ленту, закрепленную в первой и последней точках; во всех остальных точках размещены магниты. Лента притягивается к каждой точке, причем чем выше напряженность магнитного поля в точках (т. е. чем выше их кратность) , тем ближе будет притянута лента к ним. При стремлении кратности к бесконечности многочлен Безье стремится к ломаной кривой, точками сопряжения для которой служат точки-ориентиры.

10.5. НЕКОТОРЫЕ СВОЙСТВА МНОГОЧЛЕНОВ БЕЗЬЕ

Кривая, порождаемая многочленом Безье, обладает следующим интересным свойством: любую дугу, входящую в нее, также можно породить c помощью многочлена Безье [10.1]. Это свойство оказывается полезным при определении поверхностей (см. гл. 13); мы точно сформулируем и докажем это свойство для част-

Подставив $s = 2t$ в уравнение (10.23а), устанавливаем, что значения $P(t)$ при 0<=t<=0,5 определяются правой частью этого уравнения при O<=s<= 1, т. е. многочленом Безье, построенным по точкам-ориентирам P^0_0 и P^1_1. Аналогичным образом, подставив $5 = 2t—1$ в уравнение (10.23*б*), устанавливаем, что его правая часть представляет собой многочлены Безье, построенный по точкам-ориентирам P^1_1 и P^0_1; этот многочлен при изменении s от O до 1 определяет значения $P(t)$ соответствующие изменению s в диапазоне 0,5—1. Итак, для

$$\mathbf{P}_{0,\,m-1}(t) = \sum_{i=0}^{m-1} \mathbf{P}_i^i C_{m-1}^i s^i (1-s)^{m-1-i}, \qquad (10.24а)$$

$$\mathbf{P}_{1,\,m}(t) = \sum_{i=0}^{m-1} Q_i^i C_{m-1}^i s^i (1-s)^{m-1-i}. \qquad (10.24б)$$

случаяm=1 теорема доказана.

При выполнении общего шага индукции допустим, что теорема справедлива для случая m—1. Тогда можно записать, что Второй член этого уравнения, как следует из (10.26), равен

$$\mathbf{P}_{0\,m}(t) = \sum_{i=0}^{m-1} \mathbf{P}_i^i C_{m-1}^i s^i (1-s)^{m-i} + \sum_{i=0}^{m-1} \frac{\mathbf{P}_i^i + \mathbf{Q}_i^i}{2} C_{m-1}^i s^{i+1} (1-s)^{m-1-i}. \qquad (10.27)$$

для соответствующих точек Q^i_i. Нетрудно убедиться, что

$$\mathbf{Q}_i^i = \mathbf{P}_{i+1}^i, \qquad (10.25)$$

и, $\sum_{i=0}^{m-1} \mathbf{P}_{i+1}^{i+1} C_{m-1}^i s^{i+1} (1-t)^{m-(i+1)}$, (10.26)

и его можно переписать так:

$$\sum_{i=1}^{m} \mathbf{P}_i^i C_{m-1}^{i-1} s^i (1-s)^{m-1},$$

Подстановка уравнения (10.24) в уравнение (10.22а) приводит к следующему:

в результате члены уравнения (10.27) можно перегруппировать следующим образом:

$$\mathbf{P}_{0\,m}(t) = \mathbf{P}_0^0 C_{m-1}^0 (1-s)^m + \sum_{i=1}^{m-1} \mathbf{P}_i^i \left(C_{m-1}^i + C_{m-1}^{i-1}\right) s^i (1-s)^{m-i} + \mathbf{P}_m^m C_{m-1}^{m-1} s^m. \qquad (10.28)$$

Биномиальные коэффициенты в первом и третьем члене равны 1, а их сумма, входящая во второй член, равна, согласно уравнению (10. 10а), C^i_m. Таким образом, получено выражение, идентичное уравнению (10.24а), за исключением того, что m—1 заменено

на m. На этом доказательство для 0<=t<=0,5 заканчивается. Читателям в качестве упражнения предлагается провести доказательство теоремы для 0,5<=t<=1.

10.6. ДУГИ ОКРУЖНОСТЕЙ

Теперь займемся некоторыми задачами, связанными с воспроизведением кривых. В следующем разделе будут обсуждены проблемы, порождаемые дискретностью данных. Здесь рассмотрим одну специфическую задачу, которая часто возникает в прикладных задачах машинной графики: отыскание дуги окружности с заданным радиусом r, соединяющей две заданные точки с координатами x_1, y_1 (точка P_1) и x_2, y_2 (точка P_2). Для этого необходимо найти координаты центра соответствующей окружности и затем выбрать подходящую дугу из четырех возможных вариантов. В первую очередь следует определить координаты точки P_m, расположенной посередине прямой L, соединяющей две концевые точки:

$$x_m = (x_1 + x_2)/2, \tag{10 29а}$$

$$y_m = (y_1 + y_2)/2. \tag{10 29б}$$

Введем полуразности координат:

$$x_{12} = (x_1 - x_2)/2, \quad y_{12} = (y_1 - y_2)/2. \tag{10 30}$$

Центр окружности лежит на нормали к прямой L, проходящей через ее среднюю точку. Если x_{12} имеет нулевое значение (прямая L — вертикаль), то значение координаты y_c центра окружности определяется следующим образом.

$$y_c = y_m, \tag{10 31а}$$

а координата x_c центра окружности может принимать одно из двух значений:

$$x_c = x_m \pm \sqrt{r^2 - y_{12}^2}. \tag{10 31б}$$

Если прямая L не вертикальна, то для определения координат центра окружности требуется затратить несколько больше усилий. Во-первых, определяется угловой коэффициент прямой L (рис. 10.7)

$$u = \operatorname{tg} \varphi = y_{12}/x_{12} \tag{10 32}$$

и, во-вторых, — квадрат расстояния d между центром окружности и средней точкой:

$$d^2 = r^2 - (x_{12}^2 + y_{12}^2). \tag{10 33}$$

Если полученное значение d^2 оказывается отрицательным, то, следовательно, заданный радиус слишком мал и потому диаметр соответствующей окружности меньше расстояния между точками. Если оно равно нулю, то центр окружности совпадает со средней

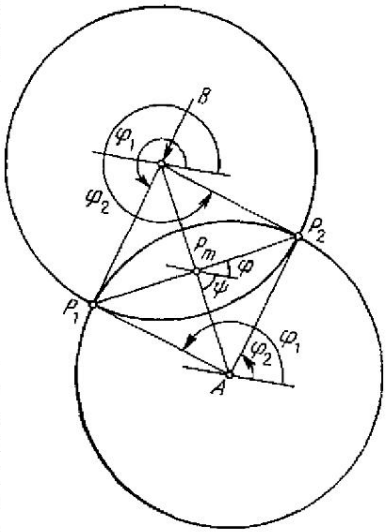


Рис 10 7 Иллюстрация к процедуре определения и некоторой дуги окружности с заданным радиусом, соединяющей две точки. Сегмент *AB* обозначает линию *L*, упоминаемую в тексте.

точкой. В остальных случаях координаты центра могут принимать значения из двух следующих множеств:

$$x_c = x_m \pm du/\sqrt{1+u^2}, \qquad (10.34а)$$

$$y_c = y_m \pm d/\sqrt{1+u^2}. \qquad (10.34б)$$

Выражения, содержащие u, представляют собой синус и косинус угла ψ (см. рис. 10.7). (Вспомните о существовании тригонометрических тождеств, определяющих синус и косинус угла через его тангенс) .Уравнения (10.34) предусматривают вычисление большего, чем необходимо, числа квадратных корней. Подстановка в них значения d из уравнения (10.33) и значения u из уравнения (10.32) дает следующее:

$$x_c = x_m \pm y_{12}\sqrt{r^2/(x_{12}^2+y_{12}^2)-1}, \qquad (10.35а)$$

$$y_c = y_m \pm x_{12}\sqrt{r^2/(x_{12}^2+y_{12}^2)-1}. \qquad (10.35б)$$

На рис. 10.7 изображены два центра окружностей. Поскольку каждому центру соответствуют также две допустимые дуги, то в общем случае задача соединения двух точек дугой с заданным радиусом имеет четыре решения. Даже если зафиксировать направление — по или против часовой стрелки, то еще остаются два решения. Предпочтение обычно отдается более короткой из двух дуг. (Если d равно нулю, то длина обеих дуг, естественно, одинакова.) Если ввести разности координат

$$x_{1c} = x_1 - x_c, \quad y_{1c} = y_1 - y_c, \qquad (10.36а)$$

$$x_{2c} = x_2 - x_c, \quad y_{2c} = y_2 - y_c, \qquad (10.36б)$$

то можно определить значения углов относительно оси x линий, соединяющих центр с каждой концевой точкой соответственно (обозначения см. на рис. 10.7).

$$\varphi_1 = \operatorname{arctg}(y_{1c},\ x_{1c}), \qquad (10.37а)$$

$$\varphi_2 = \operatorname{arctg}(y_{2c},\ x_{2c}). \qquad (10.37б)$$

Здесь арктангенс принимает значение $0—2\pi$. Если разность $\varphi = \varphi_1 - \varphi_2$ — положительна (как на рис. 10.7 для нижнего центра *A)*, то длина дуги, проведенной от первой точки ко второй по часовой стрелке, равна $r\varphi$. Если эта разность отрицательна (как на рис. 10.7 для верхнего центра *B)*, то длина дуги, проведенной

от первой точки ко второй против часовой стрелки, равна $-r \cdot \varphi$. Приведенные формулы позволяют выбирать кратчайшую дугу с заданным направлением.

Иногда мы располагаем третьей точкой P_3 и требуется выбрать дугу, проходящую ближе всех остальных к этой точке. Используя ее координаты x_3, y_3, можно вычислить ее расстояние от обоих центров и выбрать тот центр, расстояние до которого ближе всего к значению радиуса. Если точка P_3 расположена с той же стороны прямой P_1P_2, что и центр выбранной окружности, то следует выбирать большую из двух дуг, в противном случае следует выбирать меньшую. Это решение может основываться на определении проекции сегмента P_3P_c на прямую L. (P_c обозначает центр окружности — A или B на рис. 10.7).

Лемма 10.1. Пусть на плоскости заданы три точки — K, L и и M, и $(KL)_x$ — взятая с соответствующим знаком разность координат x точек K и L; аналогичным образом определены разности координат $(KL)_y$, $(ML)_x$ и т. д. В таком случае длина проекции сегмента KL на сегмент ML равна сумме $(KL)_x(ML)_x + (KL)_y(ML)_y$, деленной на длину сегмента ML.

Эта лемма легко доказывается с помощью простых геометрических построений. В нашем случае лемму можно непосредственно использовать, считая точкой K точку P_3, точкой L — центр выбранной окружности и точкой M — центр второй окружности. Поскольку длина проекции сравнивается с половиной расстояния между центрами, деление на длину можно опустить (см. задачу 10.10).

Пример 10.4. Необходимо провести дугу радиусом 10 из начала координат (0,0) в точку (5,5) таким образом, чтобы она проходила вблизи точки (3,4). Находим, что $x_m=y_m=2,5$; и=1 и d^2= 100—50/4 = 87,5, т. е. d=9,354. В таком случае использование уравнения (10.34) приводит к получению x_c = 2,5±9,354/sqrt(2), y_c = 2,5±9,354/sqrt(2); кандидатами на роль центров дуги являются точки с координатами (8,115; — 4,155) и (—4,115; 8,115). Расстояние третьей точки от первой точки-кандидата составляет 9,59, а от второй — 8,76. Значение первой ближе к значению радиуса (10), и поэтому в качестве центра выбирается точка с координатами (8,115; — 4,115). Использование уравнения (10.36) дает x_{1c} = =—8,115; (y_{1c}=4,115; x_{2c}=—3,115, y_{2c}=9,115. По таблице значений тригонометрических функций (или с помощью калькулятора) определяются углы φ_1= = 2,67 рад и φ_2=1,9 рад; таким образом, длина дуги с направлением по часовой стрелке равна (2,67—1,9)10 = 7,7. Очевидно, что дуга с направлением по часовой стрелке короче дуги с направлением против часовой стрелки, поскольку длина соответствующей окружности равна 31,4. Числитель выражения, определяемого леммой 10.1, равен (3—8,115) (—4,115—8,115) + (4 + 4,115) (8,115+4,115) = = 12,23 (8,115—5,115) =36,69. Так как это значение меньше удвоенного значения расстояния d^2, следует выбирать более длинную из двух дуг. Итак, искомая дуга должна быть проведена по часовой стрелке.

10.7. ВОСПРОИЗВЕДЕНИЕ ПРЯМЫХ ЛИНИЙ И КРИВЫХ

В общем случае математическое описание кривой, заданной на плоскости, может быть представлено или с помощью параметрического представления

$$x = X(t), \quad y = Y(t), \tag{10.38}$$

или уравнения

$$f(x, y) = 0. \tag{10.39}$$

Обычно при воспроизведении кривых удобнее пользоваться первым способом представления: в качестве параметра t берется последовательность значений $t_1, \ldots, t_i, \ldots, t_n$ и воспроизводятся пикселы, координатами которых служат округленные значения $X(t_i)$ и $Y(t_i)$. Получаемое таким образом отображение может состоять как из отдельных пикселов, так и соединяющих их линий. Решающее значение имеет выбор числа отображаемых точек. Если точки отстоят друг от друга слишком далеко, то воспроизводимая кривая будет иметь вид пунктирной линии (если воспроизводятся только отдельные точки) или многоугольника (если точки соединены линиями). Если точки расположены слишком близко друг к другу, то это может привести не только к излишним вычислениям, но и к такой кривой, отдельные части которой будут выглядеть слишком толстыми. Расстояние d между воспроизводимыми точками, соответствующими значениям t, отличающимся друг от друга на Δt, приближенно определяется формулой

$$d = \Delta t \sqrt{X'(t)^2 + Y'(t)^2}, \tag{10.40}$$

где $X'(t)$ и $Y'(t)$ —производные $X(t)$ и $Y(t)$ по параметру t. Таким образом, в различных частях кривой значения $\cdot t$ должны выбираться по-разному. К сожалению, не всегда просто вычислить значения d с помощью уравнения (10.40). Например, в случае многочленов Безье в это уравнение приходится подставлять члены уравнения (10.14). При соединении точек линиями следует избегать возникновения острых углов, что зависит от кривизны. Последняя же обратно пропорциональна расстоянию d, и, следовательно, может потребоваться иная стратегия определения расстояния между точками. Совершенно очевидно, что воспроизведение гладкой кривой, проходящей через множество точек, не сводится исключительно к отысканию уравнения, описывающего такую кривую.

Пример 10.5. Требуется воспроизвести параболу $x=t$, $y=t^2$. Вначале вычисляется расстояние между парой точек как функция параметра t. Итак,

$$\Delta x = x_{i+1} - x_i = t_{i+1} - t = \Delta t,$$

$$\Delta y = y_{i+1} - y_i = t_{i+1}^2 - t_i^2.$$

В таком случае расстояние d_i, будет определяться как

$$d_i = \sqrt{(\Delta x)^2 + (\Delta y)^2} = \Delta t \sqrt{1 + (t_{i+1} + t_i)^2},$$

т.е. оно прелставляет собой возрастающую функцию параметра t. Отметим, что в промежуточной точке сумма $t_i + t_{i+1}$ равна $y'(t)$. Если воспроизводятся только точки, то шаг по параметру $\cdot t$ следует выбирать обратно пропорциональным значению t.

10.7.1. ВОСПРОИЗВЕДЕНИЕ КРИВЫХ НА ОСНОВЕ ИСПОЛЬЗОВАНИЯ ДИФФЕРЕНЦИАЛЬНЫХ УРАВНЕНИЙ

Кривые, представленные уравнением вида $f(x, y)=0$, могут воспроизводиться аналогично кривым с параметрическим представлением, если это уравнение удается разрешить относительно одной из переменных. Другой подход к представлению таких кривых основывается на дифференцировании уравнения (10.39). Поскольку $f(x, y)$ — константа, ее полный дифференциал равен нулю,

$$\frac{\partial f}{\partial x}\,dx+\frac{\partial f}{\partial y}\,dy=0. \tag{10.41}$$

Введя обозначения $F_x(x, y)=df/dx$, $F_y(x, y)=df/dy$, получаем

$$dy/dx=-F_x(x,\ y)/F_y(x,\ y). \tag{10.42}$$

и, следовательно,

Искомая кривая представляет собой решение последнего дифференциального уравнения. Это обстоятельство лежало в основе широко использовавшегося метода воспроизведения кривых на аналоговых устройствах, поскольку многие уравнения такого рода моделируются с помощью достаточно простых электронных схем. Дискретную аппроксимацию указанного уравнения можно задать следующим образом:

$$x_{k+1}-x_k=-c\,F_y(x_k,\ y_k), \tag{10.43а}$$

$$y_{k+1}-y_k=c\,F_x(x_k,\ y_k), \tag{10.43б}$$

где c — произвольная константа. Начнем с точки $(x_0,\ y_0)$, тогда остальные точки можем определить с помощью уравнения (10.43). Очевидно, что константа c определяет плотность получаемых точек. В результате появляются те же проблемы, что и при построении кривой, задаваемой уравнением (10.38). В данном случае, однако, возникает новое существенное препятствие. Так как уравнение (10.43) является лишь некоторой аппроксимацией уравнения (10.42), может оказаться, что построенная кривая будет существенно отличаться от искомой. Полное исследование этой проблемы включает рассмотрение устойчивости численных методов решения и выходит за пределы задач нашей книги (см. разд. 10.9). Следующий пример иллюстрирует эту проблему и способ ее разрешения в простом случае.

Пример 10.6. Уравнение окружности с радиусом r и центром в начале координат имеет вид $x^2+y^2=r^2$, так что уравнения (10.43) принимают следующий вид

$$x_{k+1}=x_k-2cy_k, \tag{10.44а}$$

$$y_{k+1}=y_k+2cx_k. \tag{10.44б}$$

Если требуется, чтобы исходная точка имела координаты $y_0=r$ и $x_0=0$, то можно показать, что приведенные уравнения в конечных разностях имеют решение

$$x_k=r\,(\sqrt{1+4c^2})^k\sin(k\varphi), \tag{10.45а}$$

$$y_k=r\,(\sqrt{1+4c^2})^k\cos(k\varphi), \tag{10.45б}$$

где ф — угол, тангенс которого равен 2c[1]. Отметим, что значения x и y пропорциональны некоторой степени числа, большего единицы, а именно значения квадратного корня суммы $1+4c^2$. Следовательно, их значения неограниченно возрастают при увеличении значения k и полученная кривая окажется не окружностью, а спиралью, удаляющейся от центра, т. е. явно не той кривой, которую требовалось построить. Выберем теперь число b, обладающее тем свойством, что $b^2+4c^2=1$, и заменим уравнения (10.44) следующими:

$$x_{k+1} = bx_k - 2cy_k, \tag{10.46а}$$

$$y_{k+1} = by_k + 2cx_k, \tag{10.46б}$$

Выбор такого параметра 6 оказывается возможным при c<1/2. Тогда решение этих разностных уравнений принимает вид

$$x_k = r \sin(k\varphi), \tag{10.47а}$$

$$y_k = r \cos(k\varphi), \tag{10.47б}$$

где тангенс угла φ равен $2c/b$. Очевидно, что точки, определяемые уравнениями (10.47), лежат на окружности и длина дуги, заключенной между двумя соседними точками, равна rφ.

Значение константы c следует выбирать таким образом, чтобы получаемые точки располагались достаточно близко друг к другу. В предыдущем примере определяющим фактором является отношение $2c/sqrt(1—4c^2)$

10.7.2. ВЛИЯНИЕ ОШИБОК ОКРУГЛЕНИЯ НА ВОСПРОИЗВЕДЕНИЕ КРИВЫХ

Выбор надлежащего шага дискретизации кривой не является единственной проблемой, возникающей при попытках добиться хорошего воспроизведения. Элементы структуры воспроизводимой кривой испытывают также влияние ошибок округления значений $X(t)$ и $Y(t)$. В самом деле, все воспроизводимые изображения характеризуются некоторой конечной разрешающей способностью и для их получения каждый раз должен выбираться тот пиксел, координаты которого лежат ближе всего к вычисленным значениям координат, и воспроизводиться должна точка именно с этими координатами, а не с координатами точно $X(t)$ и $Y(t)$ (или x_k и y_k если используется метод, описанный в предыдущем подразделе). Некоторые из возникающих при этом проблем были рассмотрены в разд. 7.6 в рамках задания прямых на некоторой сетке. Пусть требуется воспроизвести изображение прямой, связывающей точку (x_1, y_1) с точкой (x_2, y_2). Точки этой прямой определяются уравнением (10.2), которое можно использовать для получения значений y при разных значениях x.

Пример 10.7. Пусть $x_1=y_1=0$ и $x_2 = 40$, $y_2=17$. При этом уравнение (10.2) принимает вид $y=17x/40$. Используя округление до ближайшего целого числа, получаем значения координат, образующих н-связный маршрут (результаты

[1] Читатели, не знакомые с теорией уравнений в конечных разностях, могут подставить уравнения (10.45) в уравнения (10.44) и убедиться в том, что обе части имеют одно и то же значение В процессе проверки следует воспользоваться тем, что $sqrt(1+4c^2) = 1/\cos\varphi$.

Таблица 10.1

Сравнение точных и округленных значений координат

Значение x	Значение y		Значение x	Значение y	
	точное	округленное		точное	округленное
1	0,425	0	5	2,125	2
2	0,85	1	6	2,55	3
3	1,275	1	7	2,975	3
4	1,7	2	. . .	. . .	. . .

сведены в табл. 10.1). Если угловой коэффициент прямой больше единицы, в качестве независимой переменной следует использовать y.

Этот пример показывает, что прямая, построенная по округленным значениям y, имеет вид «лесенки» и такое изображение может оказаться неприемлемым с эстетической точки зрения. Как отмечалось в разд. 7.6, если C — точка, расположенная на прямой между точками A и B, воспроизводимая прямая, соединяющая точки Л и С, может не являться частью прямой AB, что и было продемонстрировано в примере 7.1. Если используемое устройство отображения позволяет воспроизводить тоновые изображения, то не обязательно воспроизводить все точки с одинаковой яркостью. В точках, в которых точные и округленные значения близки, как, например, в точках $x = 2$, 5, 7 и т. д., воспроизводится один пиксел с максимальным уровнем яркости. В остальных точках воспроизводятся два пиксела с пониженной яркостью. В результате воспроизводимая прямая имеет более гладкий вид. Этот метод можно использовать для воспроизведения произвольных кривых. Исчерпывающий обзор подобных методов выходит за пределы задач нашей книги (см. разд. 10.9). Читатель должен также иметь в виду, что при решении многих прикладных задач, например при фотонаборе, следует воспроизводить двухуровневое изображение.

При вычислениях, связанных с воспроизведением изображения, в процессоре дисплея могут возникать дополнительные осложнения. Поскольку такие процессоры обычно оснащаются арифметическими устройствами для работы с целыми числами, которые характеризуются низкой точностью, коэффициенты математического описания кривой могут подвергнуться столь радикальному округлению, что построенные кривые могут не проходить даже через те точки, о которых известно, что они лежат на искомой кривой.

Пример 10.8. Требуется соединить точку (0,0) с точкой (17,11) дугой окружности, центр которой расположен на оси x. Уравнение, описывающее такую окружность, имеет вид $x^2 - 2x_c x + y^2 = 0$, и мы легко находим, что $2x_c = 410/17 = 24{,}1176...$ Это значение округляется до 24, и для построения дуги следует использовать уравнение $F(x, y) = x^2 - 24x + y^2 = 0$. Поскольку это уравнение даёт $F(17,11)=2$, оказывается, что концевые точки не лежат на построенной кривой.

Нет ничего страшного в том, что концевая точка не лежит на воспроизводимой кривой, если эта точка попадает на воспроизводимые пикселы: однако последнее полностью зависит от того, ка-

кое применяется конкретное правило вычерчивания кривой (см. задачу 10.9). Таким образом, не следует использовать критерий равенства координат для принятия решения об окончании вычерчивания кривой прежде, чем будет проведен тщательный анализ, подтверждающий, что в результате не возникнут серьезные проблемы. Другой критерий остановки предусматривает проверку равенства лишь по одной координате, после чего проверяется, не превышает ли отклонение по другой координате единицу. (К сожалению, в ряде стандартных программ для вычерчивания кривых второго порядка используется критерий равенства и порождение пикселов воспроизводимой кривой продолжается до тех пор, пока она не пройдет через заданную концевую точку.)

10.8. ПРОГРАММА-РЕДАКТОР ТОЧЕК

При построении кривых по точкам в задачах машинной графики требуется такой набор параметров, чтобы полученная на их основе кривая имела заданную форму. Часто очень трудно заранее выбрать подходящие значения этих параметров, поэтому приходится работать в интерактивном режиме. При использовании многочленов Безье необходимы средства для ввода множества точек-ориентиров, определения и воспроизведения многочлена. Если форма полученной кривой нас не устраивает, мы должны иметь возможность изменить размещение точек-ориентиров, ввести новые точки-ориентиры или удалить какие-то из старых. Основным инструментом для осуществления подобных операций служит программа интерактивного редактирования графической информации — диалоговый графический редактор. Мы будем называть его программой-редактором точек (чтобы отделить его от программ-редакторов изображений или текстов). Программу-редактор точек можно применить в целом ряде задач, помимо построения кривых по точкам, а его рассмотрение в данной главе не следует считать признаком того, что определение многочленов Безье представляет основную область использования этого редактора.

При разработке программы-редактора точек возникают две основные проблемы: структура данных, используемая для хранения точек, и средства ввода и адресации (указывания) точек.

10.8.1. СТРУКТУРА ДАННЫХ ДЛЯ ПРОГРАММЫ-РЕДАКТОРА ТОЧЕК

Простейшей из возможных структур является двухмерный массив координат *x, y* точек. Однако ее использование приводит к тому, что операции ввода и удаления становятся очень громоздкими. Лучше применять связный список. Элементы такого списка состоят из четырех компонент: координаты *x,* координаты *y,* адреса предшествующего элемента списка *(p)* и адреса следующего элемента списка *(n)*. Эту структуру можно реализовать в виде четы-

рех одномерных массивов, причем в качестве адреса используется индекс массива.

Пример 10.9. Пусть исходными служат точки (5,8), (11,22) и (14,18) В таком случае элементы списка имеют вид

(5,8,0,2), (11,22,1,3), (14,18,2,0).

Здесь 0 обозначает указатель несуществующего элемента — именно таким образом отмечаются первая и последняя точки списка (рис. 10.8,*а*). Если ввести точку (9,10) между первой и второй точками, то список принимает вид

(5,8,0,4), (11,22,4,3), (14,18,2,0), (9,10,1,2)

(рис 10.8,*б*). Удаление последней точки последовательности (третьей в списке) приводит к следующему преобразованию списка.

(5,8,0,4), (11,22,4,0), (14,18,3,3), (9,10,1,2)

(рис. 10.8,*в*); здесь указание удаляемой точки содержится в описании самой точки. Взаимная перестановка второй и последней точек приводит к следующему преобразованию списка.

(5,8,0,2), (11,22,1,4), (14,18,3,3), (9,10,2,0)

(рис. 108,*г*).

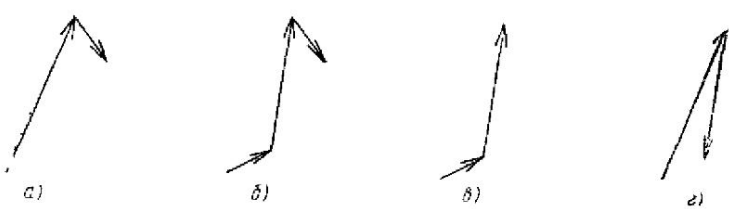


Рис. 10.8 Расположения точек, рассмотренные в примере 10.9

Еще лучший способ реализации связного списка обеспечивается использованием массивов структур и указателей В принципе, оперирование данными при этом производится так же, как в приведенном случае, однако описание алгоритма упрощается. Определим структуру *точка* следующим образом:

структура *точка* = {*x, y, p, n*}.

(Предполагается, что читатель знаком со структурами и указателями, используемыми во многих языках программирования.) Точки воспроизводимой кривой заносятся в массив, состоящий из таких структур. Мы, как и прежде, будем применять символы О (указание на несуществующий элемент) и *L* для обозначения первой свободной ячейки памяти, которую можно занять; символ *q* обозначает указатель текущей ячейки памяти[1]. Таким образом, для воспроизведения линии между текущей и предыдущей точка-

[1] Запись (*q->u* используется для обозначения объекта массива структур *u*, который обозначается указателем *q*.

ми можно воспользоваться следующими командами (см. разд. 1.7, табл 1.1):

setp((q->p)->x, (q->p)->y),

vec(q->x, q->y).

Первая команда обеспечивает вывод текущей точки на графопостроитель с координатами, определяемыми точкой, предшествующей точке, определенной указателем *q*. Вторая команда обеспечивает соединение двух точек вектором.

Пример 10.10. Следующая последовательность операций обеспечивает введение в список точки, причем пользователь указывает как позицию, в которую точка вводится, так и местоположение новой точки.

1. *readp* (x_1, y_1). (Пользователь указывает на экране точку.)
2. Определение такого указателя *q*, что пара (*q->x, q->y*) оказывается ближайшей к точке (x_1, y_1).
3. $q_n = q->n$. (Сохранение адреса старой «следующей точки»)
4. *g->n=L.* (Новая «следующая точка» помещается в первой свободной позиции.)
5. Указание пользователя.
6. *readp* (x_2, y_2). (Задание второй позиции на экране.)
7. $L->x=x_2$, $L->y=y_2$. (Добавление к списку точки.)
8. $L->p=q, L->n=q_n$ (Связь.)
9. $q_n->пред = L$. (Коррекция связи старой «следующей точки»)
10. Увеличение *L*. (Коррекция адреса первой свободной позиции)

При решении многих прикладных задач важно оперировать с группами точек, например с дугами. Для этого можно воспользоваться какой-нибудь иерархической структурой данных. Кривую можно представить с помощью указателя массива, содержащего точки этой кривой, и т. д.

Лучше писать программы, реализующие редактор точек, не ориентируясь на конкретное устройство отображения. В частности, ведущая программа должна иметь дело лишь с адресами (парами координат *x, y),* командами (используя общие названия) и, естественно, именами файлов. Все процедуры, ориентированные на конкретное устройство отображения, должны быть выделены и явно определены как предназначенные для конкретной аппаратной реализации. (Более подробные сведения приведены в следующем подразделе.) Ниже приводится перечень типичных команд. Для выполнения каждой из перечисленных команд необходимо либо множество точек (число последних указывается в скобках), либо имя файла (в скобках помещается буква *ф).* Буквы, набранные жирным шрифтом, обозначают сокращенные названия команд.

plot(0) Воспроизвести точки, находящиеся в списке.

bezier (0) Воспроизвести многочлен Безье, используя в качестве точек-ориентиров элементы списка.

append (0) Ввести новые точки в конец списка или завести новый список, если текущий список пустой.

insert (1) Ввести в список точки, поместив их перед точкой с указанным адресом.

delete (1) Удалить точку из списка.

repeat (1) Ввести в список новую точку с координатами, идентичными координатам точки с указанным адресом. (Эту команду можно использовать для увеличения кратности точек-ориентиров.)

move (2) Переместить адресованную точку в позицию, соответствующую второму адресу.

write (ф) Записать содержимое списка в файл с соответствующим именем.

read (ф) Произвести выборку содержимого файла и записать его в список.

Введение команд *insert* и *append* предполагает, что пользователь предоставляет последовательность точек, добавляемых к списку. Необходима также специальная команда *escape,* указывающая, что процесс закончен.

10.8.2. ВВОД И ВЫВОД ПРИ ИСПОЛЬЗОВАНИИ ПРОГРАММЫ-РЕДАКТОРА ТОЧЕК

Координаты точек можно вводить в редактор с помощью функции *readp (x, y),* описанной в разд. 1.7 (см. табл. 1.1). Прежде чем выбрать команду или адрес, необходимо выполнить еще ряд операций. В некоторых устройствах, предназначенных для указания точки на экране дисплея с автоматическим вводом ее координат, предусмотрены клавиши (кнопки), позволяющие пользователю вводить помимо координат один или несколько символов. Если число таких символов достаточно велико, то их можно использовать для кодирования команд. В противном случае пользователь может обратиться к *меню* — области, выделенной на боковой части экрана дисплея и разделенной на отдельные участки, обозначенные названиями соответствующих команд. Когда значения координат *x, y,* сообщенные программе, попадают в один из этих участков, выбирается соответствующая команда.

Вторая проблема связана с выбором в списке точки, расположенной ближе всего к указанной пользователем. (Это соответствует случаю, когда пользователь хочет выбрать одну из имеющихся точек.) Тривиальным решением является поиск минимального эвклидова расстояния между заданной точкой и точками списка. Более эффективный способ предусматривает организацию вспомогательной структуры данных, в которой точки упорядочены в соответствии с их местоположением.

Алгоритм 10.3 представляет собой минимальный вариант программы-редактора точек. При его разработке предполагалось, что кроме элементарных команд преобразования графических данных, приведенных в табл. 1.2, предусмотрены также следующие элементарные команды:

command (x, y, c), обеспечивающая вывод цепочки *c,* содержащей название подпрограммы, или пустой цепочки, если точка *(x, y)* не указывает ни на какую команду.

place (x, y, q), обеспечивающая обнаружение точки списка, расположенной ближе всех точек списка к точке *(x, y); q* — значение указателя, соответствующее обнаруженной точке.

Алгоритм 10.3. Программа-редактор точек.

J. Repeat (неограниченно!) шаги 2—5.
   Begin.
2. *readp (x, y).*
3. Обращение к *command (x, y, c).*
4. If цепочка *c* — пустая или если для выполнения подпро
   граммы *c* необходим адрес, then производится обраще
   ние к *place (x, y, q).*
5. Обращение к *c.*
   End.
6. Конец алгоритма.

Алгоритм 10.3. предполагает, что имеется команда *quit,* обеспечивающая выход из цикла. Значения *x и y,* получаемые на шаге 2, должны быть введены в стек для дальнейшего использования.

10.9. БИБЛИОГРАФИЧЕСКИЕ ЗАМЕЧАНИЯ

Методы интерполирования с помощью многочленов описываются в большинстве учебников по численным методам (например, в [10.6]). Имеются книги, непосредственно посвященные этой теме (например, [10.5]). Обстоятельное математическое исследование этой задачи можно найти в монографии [10.10]. Многочлены Безье обязаны своим именем французскому математику, предложившему их в процессе исследования, выполнявшегося по заказу автомобильной компании «Рено» [10.2]. Формулы, описывающие дуги окружностей, относятся к стандартному аппарату аналитической геометрии и приведены здесь в связи с широким интересом, проявляемым к их использованию.

Раздел, посвященный воспроизведению кривых, мог бы иметь значительно больший объем, так как практическое построение кривых на дискретной сетке — это, действительно, увлекательная задача. Дополнительную информацию о воспроизведении дуг читатель может найти в статьях [10.2, 10.7], а сведения об использовании шкалы серого тона для улучшения векторных изображений при работе на растровых графических устройствах — в статьях [2.2,2.7, 10.4, 10.11].

Проблемы устойчивости численных методов решения обсуждаются в большинстве учебников по численным методам. В частности, проблемы расходимости, обусловленной дискретной аппроксимацией, могут изучаться с помощью методов, представленных в литературе по численным методам решения дифференциальных Уравнений *и* по уравнениям в конечных разностях (например, [10.8]).

Программа-редактор точек, описанная в данной главе, обладает очень умеренными возможностями — большинство систем обработки графической информации предлагают пользователю много больше. Статья [10.3] посвящена обзору проблемы редактирования.

10.10. ЗАДАЧИ

10.1. Через множество точек $\{x_i, y_i\}$ на плоскости требуется провести гладкую кривую. Предлагается следующий метод решения. Рассмотрим заданные точки как вершины многоугольника и для каждого значения i построим биссектрису угла при соответствующей вершине. Будем считать нормаль к биссектрисе, проходящую через вершину, касательной к искомой кривой в данной вершине и воспользуемся уравнением (10.5) для определения дуги третьего порядка, соединяющей каждую пару точек.

а. Пусть NX_i обозначает разность $x_i — x_{i-1}$, разделенную на длину вектора, проведенного из точки (x_{i-1}, y_{i-1}) в точку (x_i, y_i), и пусть Ny_i обозначает соответствующую разность по координате y. Требуется показать, что нормаль к биссектрисе в точке (x_i, y_i) определяется уравнением

$$(Nx_{i+1} — Nx_i)(x — x_i) + (Ny_{i+1} — Ny_i)(y — y_i) = 0,$$

б Подставьте последнее выражение в уравнение (105) и получите представление дуги третьего порядка в явном виде. Решите эту задачу для задания кривой уравнением $y=f(x)$ и параметрического задания $x = x(t)$ и $y = y(t)$ при $0<=t<=1$.

в. Проанализируйте преимущества и недостатки обеих форм представления кривой и выведите условие, определяющее вид многоугольника таким образом, чтобы гладкая кривая, заданная с помощью параметрического представления, не пересекала этот многоугольник.

102. Выведите формулу (10.10а)

10.3. Напишите программу, реализующую алгоритм 10 1.

10.4. Напишите программу, реализующую алгоритм 10.2, и сравните время ее прогона с временем прогона программы предыдущей задачи. Постройте график зависимостей этих времен от m. Как реально затраченное время соотносится с результатами теоретического прогноза? Чем Вы объясните их расхождение?

10.5. Докажите вторую часть теоремы 10.1 (для случая $0,5<=t<=1$).

10.6. Напишите простую программу-редактор и предусмотрите в ней команду, обеспечивающую при необходимости возможность построения многочлена Безье (Можете воспользоваться одной из программ, написанных пои решении задач 10.3 и 10.4) Задайте множество точек, образующих контур физического объекта, и попробуйте построить многочлен Безье, проходящий вблизи заданных точек. Строить этот многочлен придется, используя интерактивный режим работы, поскольку выбор точек-ориентиров отнюдь не очевиден. Следовательно, указанный редактор составляет существенную часть задания.

Замечание. Старайтесь организовать программу таким образом, чтобы в будущем ее можно было пополнить новыми возможностями.

107 Сравните число бит, необходимое для хранения коэффициентов многочлена Безье, аппроксимирущего контур объекта, с числом бит, необходимым для хранения описания этого же контура, закодированного с помощью дифференциального цепного кода. Повторите это сравнение для других параметров. Можно ли на этом основании сделать какие-либо общие выводы?

10.8. По заданным параметрическому описанию кривой $X(t)$, $Y(t)$ и N — числу выборочных точек напишите процедуру выбора плотности размещения выборочных точек на сегменте [0,1], обеспечивающей максимально возможную

гладкость кривой, воспроизводимой посредством векторов, соединяющих N выборочных точек.

10.9. Требуется воспроизвести функцию ax^n на экране растрового графического устройства. Поскольку а не является целым числом, а нам желательно использовать для вычислений лишь арифметическое устройство, предназначенное для работы с целыми числами, будем воспроизводить функцию bx^n, где b-—целое число, ближайшее к a. Предполагается, что a и b больше 1; то же относится к числу п. Чтобы получить непрерывную кривую, необходимо использовать для представления каждого значения x более одного пиксела. Можно ли сформулировать правило, всегда обеспечивающее представление округленного значения ax^n с помощью такого пиксела? Если Вы считаете это невозможным, дайте формальное доказательство своего утверждения.

10.10. Напишите программу, обеспечивающую проведение прямой через пару точек и не использующую ни одной графической команды. Сделайте то же самое для окружности.

10.11 Покажите, что окружность, заданная неколлинеарными точками (x_1, y_1), (x_2, y_2) и (x_3, y_3), имеет центр в точке (x_c, y_c), определяемой следующими уравнениями:

$$x_c = \frac{C_1 (y_2 - y_3) - C_2 (y_1 - y_2)}{(x_1 - x_2)(y_2 - y_3) - (x_2 - x_3)(y_1 - y_2)}, \qquad (10.48а)$$

$$y_c = \frac{C_2 (x_1 - x_2) - C_1 (x_2 - x_3)}{(x_1 - x_2)(y_2 - y_3) - (x_2 - x_3)(y_1 - y_2)}, \qquad (10.48б)$$

где $C_1 = (x_1 - x_2)(x_1 + x_2) + (y_1 - y_2)(y_1 + y_2)$; $C_2 = (x_2 - x_3)(x_2 + x_3) + (y_2 - y_3) * *(y_2 + y_3)$.

Покажите, что в уравнениях (10.48) знаменатель равен 0, если соответствующие три точки коллинеарны.

Глава 11

ПОСТРОЕНИЕ КРИВЫХ ПО ТОЧКАМ С ПОМОЩЬЮ СПЛАЙНОВ

11.1. ВВЕДЕНИЕ

При решении многих прикладных задач, предусматривающих построение кривых по точкам, может потребоваться внести изменения в одну часть кривой, не затрагивая другие. Мы будем считать схему локальной, если локальные изменения не распространяются на другие части изображения. Очевидно, что многочлены, рассмотренные в разд. 10.2, свойством локальности не обладают, а многочлены Безье обладают лишь квазилокальностью. Изменение расположения или кратности одной из точек-ориентиров требует пересчета всей кривой, даже несмотря на то, что указанные изменения могут слабо влиять при удалении от соответствующей точки-ориентира. Кусочно-полиномиальные функции несомненно являются средством реализации локальных изменений. Сначала рассмотрим такие функции применительно к представлению кривой в виде $y = y(x)$, а затем — применительно к параметрическому

представлению. В общем виде кусочно-полиномиальные функции представляются следующим образом:

$$p(x) = p_i(x), \quad x_i \leqslant x \leqslant x_{i+1}, \quad i = 0,1,.., k-1; \tag{11.1а}$$

$$p_i^{(j)}(x_i) = p_{i+1}^{(j)}(x_i), \quad j = 0, 1, .., r-1;$$

$$i = 1,..., k-1, \tag{11.1б}$$

Точки $x_1, \ldots, x_{k-1}$, которые делят сегмент $[a, b]$ на k подсегментов, обычно называют точками склеивания, а точки воспроизводимой кривой, соответствующие этим значениям x, — узлами. Для удобства будем использовать обозначения $x_0 = a$ и $x_k = b$. Функции $p_i(x)$ представляют собой многочлены со степенью не выше m. Условия непрерывности в точках склеивания задаются второй группой уравнений, в которых $p_i^{(0)}(x)$ обозначает $p_i(x)$ и $p_i^{(j)}$ — j-ю производную $p(x)$ ($j>0$). Иногда отсутствие ограничений указывается заданием $r=0$. При $r=1$ речь идет о непрерывной функции, на производные которой никаких ограничений не наложено. Если $r=m+1$, то сегмент $[a, b]$ покрывается одним многочленом, и, следовательно, $r=m$ — максимальное число ограничений, порождающих нетривиальную кусочно-полиномиальную функцию. Случай, -когда $r = 3$ и $m = 3$, имеет особое историческое и практическое значение, поскольку именно для обозначения соответствующих кусочно-полиномиальных функций был впервые предложен термин «сплайн».

Введение в теорию сплайнов представлено в разд. 11.2, а основные сведения о B-сплайнах— в разд. 11.3. В разд. 11.4 и 11.5 •обсуждаются соответствующие вычислительные проблемы, а в разд. 11.6 и 11.7 демонстрируются преимущества использования сплайнов в машинной графике.

11.2. ОСНОВНЫЕ ОПРЕДЕЛЕНИЯ

Происхождение термина «сплайн» относится к тому времени, когда еще не существовала машинная графика и чертежник, чтобы провести гладкую кривую через заданные точки, часто пользовался грузиками, помещая их в заданных точках и придавая с их помощью искомую форму гибкой деревянной линейке, называемой сплайном. Эти грузики имели выступ, который помещался в прорезь сплайна, приклепляя его к данной точке, но позволяя поворачиваться относительно нее. Если обратиться к теории упругости, то можно доказать, что результирующая кривая представляет собой (приближенно) кусочный кубический многочлен, являющийся непрерывным и имеющий непрерывные первую и вторую производные. Эти условия гарантируют также, что кривая имеет постоянную кривизну и разрывы возникают лишь в третьей производной. Поскольку человеческому глазу чрезвычайно трудно уловить последние, результирующая кривая выглядит совершенно гладкой. Если вдоль сплайна совершается механическое движение, то непрерывность второй производной предполагает непрерывность ус-

корения и, следовательно, отсутствие резких изменений приложенной силы. Эти два свойства делают весьма целесообразным применение сплайнов при решении многих прикладных задач. Однако существуют случаи, когда достаточно использовать меньшее число условий непрерывности или многочлен более низкой степени — тем не менее термин «сплайн» используется и в этих случаях. В сущности, пока относительно употребления этого термина наблюдается мало согласованности. В данной книге используется следующая терминология.

Определение 11.1. *Простым сплайном* называется кусочно-полиномиальная функция, задаваемая уравнениями (11.1) прн $r = m$. Термины «линейный сплайн», «квадратичный сплайн» и «кубический сплайн» относятся к указанной кусочно-полиномиальной функции при $m = 1$, 2 и 3 соответственно.

Определение 11.2. *Сплайном* называется кусочно-полиномиальная функция, задаваемая уравнениями (11.1) при $r<m$.

Можно отметить, что функции, использованные в пределах 10.2 (см. рис. 10.1,*б*) и 10.3 (см. рис. 10.2), являются квадратичными сплайнами, а кривая из задачи 10.1—кубическим сплайном (см. также задачу 11.1). Очевидно, что можно использовать интерполяционные, а также аппроксимационные сплайны. В обоих случаях можно знать число степеней свободы кривой. Из уравнений (11.1) следует, что для $k(m+1)$ коэффициентов многочлена минус $(k—1)r$ ограничений общее число степеней свободы некоторого сплайна составляет $k(m—r)+k + r$. Простой сплайн, для которого $m = r$, имеет всего лишь $k + m$ степеней свободы.

Большинство работающих в данной области согласны с тем, что важнейшим моментом, от которого зависит успешное использование сплайнов, является выбор числа и расположения точек склеивания. Убедиться в важности этого выбора читатели могут, обратившись к примерам разд. 10.2. Оказывается, что это — задача значительно более трудная, чем определение интерполяционного или аппроксимационного сплайна после выбора точек склеивания. Обсуждение этой проблемы в подробностях будет отложено до следующей главы, здесь мы ограничимся лишь некоторыми предварительными замечаниями. Отметим, что если допустить слияние точек склеивания, т. е. если $x_i = x_{i+1}$, то число ограничений автоматически уменьшается на 1. При кратности точек склеивания, равной $r+1$, снимаются все ограничения. Введение кратных точек склеивания, позволяющее порождать не простые сплайны, обладает как теоретическими, так и практическими преимуществами, и на некоторых из них мы остановимся ниже. Многие авторы пользуются терминами «сплайн с простыми узлами» и «сплайн с кратными узлами» для обозначения простых сплайнов и просто сплайнов соответственно. Кроме того, существуют задачи, при решении которых удобнее, чтобы значения r и m отличались друг от Друга, причем, быть может, в каждой точке склеивания.

Еще одна важная проблема, возникающая при использовании сплайнов, связана с формой математического описания кривой. В

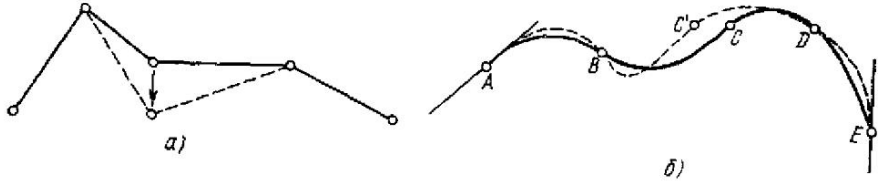
уравнение (11.1) входит слишком много параметров и использование ограничений для их исключения выливается в довольно трудоемкий анализ. (Можно показать, что эти ограничения на самом деле линейно независимы и, следовательно, каждое из них эффективно.) Другую форму описания кривой задает следующее уравнение:

$$p(x) = p_1(x) + \sum_{i=1}^{k-1} q_i (x - x_i)_+^m, \qquad (11.2)$$

где функция t^m+ принимает значение t^m при $t>0$ и 0 при $t<=0$. Отметим, что функция t^m и ее первые $(m-1)$ производных равны нулю при $t = 0$, т. е кусочно-полиномиальная функция $p(x)$ и ее первые $(m-1)$ производных непрерывны во всех точках склеивания. Член q_i пропорционален размеру скачка в точке разрыва x_i m-й производной, и, следовательно, функция, определяемая уравнением (11.2), обладает теми же свойствами, что и функция, определяемая уравнениями (11.1) при $r=m$. В частности,

$$p_j(x) = p_1(x) + \sum_{i=1}^{j} q_i (x - x_i)_+^m. \qquad (11.3)$$

Уравнение (11.2) имеет лишь $(m+1) + (k-1)=m + k$ свободных параметров, т. е. минимальное их число. Его форма близка к форме многочлена, что делает его использование крайне удобным во многих теоретических исследованиях. К сожалению, уравнение (11.2) обладает и серьезными недостатками. Во-первых, из уравнения (11.3) следует, что при необходимости изменить функцию на подсегменте, т. е. значение q_i в его левой концевой точке, представление этой функции должно быть изменено на всех последующих подсегментах. Однако, как показывает уравнение (11.1), это совсем не обязательно. Если, например, $m=1$, то вид $p(x)$ приходится изменять лишь на двух подсегментах (рис. 11.1,*а*). Подобным образом, при $m = 2$ число подсегментов не должно превышать четырех. Действительно, при рассмотрении примера, приведенного на рис. 11.1,*б*, можно было бы зафиксировать (и не изменять больше) положения узлов A и • и проходящих через них касательных, отделив таким образом от изменяемой части сплайна его остальную часть. Если положения узлов B и D (но



Тис. 11.1. Иллюстрация локальных свойств сплайнов:
а — при $m=1$ изменение положения одною из узлов требует пересчета значений сплайна лишь на двух смежных подсегментах, *б* — при $m=2$ пересчет должен быть проведен на четырех подсегментах

не касательных в них) также фиксируются, то параболические дуги, из которых состоит сплайн между узлами A и B, D и E, полностью определены. В таком случае сплайн с узлами в точках B, C' и D характеризуется в соответствии с заданием тремя узлами, двумя касательными в концевых точках и условием непрерывности в точке C' — всего шестью ограничениями, причем число степеней свободы также равно шести. Поскольку взаимная связь представлений на отдельных сегментах реализуется через ограничения, задаваемые в точках склеивания, можно полагать, что в общем случае число затрагиваемых сегментов должно быть пропорционально числу ограничений. Следующее утверждение выражает это допущение в строгом виде для случая, когда в подсегмент (x_i, x_{i+1}) вводится новая точка склеивания.

Утверждение 11.1. Если в подсегмент (x_i, x_i+1) вводится новая точка склеивания, то соответствующий сплайн необходимо изменять лишь на подсегменте (x_i, x_{i+m}).

Доказательство. Пусть u — число затронутых (исходных) подсегментов. Согласование значений сплайна и его первых m—1 производных в точках x_i и x_{i+u} требует введения 2т ограничений. Общее число подсегментов, в которые могут вводиться изменения, равно $u+1$, а полное число степеней свободы— $(m + u+1)$. Следовательно, u —m—1.

При $m=1$ не требуется проводить изменения на другом подсегменте, но при $m = 3$ два последующих подсегмента могут оказаться подлежащими изменению. Если число подсегментов не изменяется, но на один из них вводится дополнительное ограничение, то с помощью аналогичных рассуждений можно показать, что $u = m+1$. Читателя может удивить, что после проведения изменений модифицируются лишь подсегменты, а не наоборот. Это недоумение сразу разъяснится, если разделить изменения собственно кривой и изменения представления кривой. Так, утверждение 11.1 относится к изменениям, производимым в представлениях кривых.

Второй недостаток уравнения (11.2) состоит в том, что при его использовании некоторые задачи аппроксимации обнаруживают тенденцию к неустойчивости в случае решения их численными методами Поэтому при решении многих практических задач используется третья форма представления сплайнов: они представляются в виде сумм других сплайнов, в частности сплайнов специального вида — B-сплайнов.

11.3. В-СПЛАЙНЫ

B-сплайн — это сплайн, равный нулю на всех подсегментах, за исключением $m + 1$. На рис. 11.2 приведены примеры линейного, квадратичного и кубического В-сплайнов. Эти сплайны можно определить рекурсивно следующим образом.

Определение 11.3. Б-сплайн с постоянным значением на i-м подсегменте задается выражением

$$N_{i,\,0}(x) = \begin{cases} 1, & x_i \leqslant x \leqslant x_{i+1}; \\ 0, & \text{в остальных случаях.} \end{cases} \qquad (11.4)$$

B-сплайн m-й степени на сегменте $[x_i,\ x_{i+m+1}]$ определяется выражением

$$N_{i,\,m}(x) = \frac{x - x_i}{x_{i+m} - x_i} N_{i,\,m-1}(x) + \frac{x_{i+m+1} - x}{x_{i+m+1} - x_{i+1}} N_{i+1,\,m-1}(x). \qquad (11.5)$$

Уравнения (11.4) и (11.5) можно использовать для задания в явно виде B-сплайнов низших степеней.

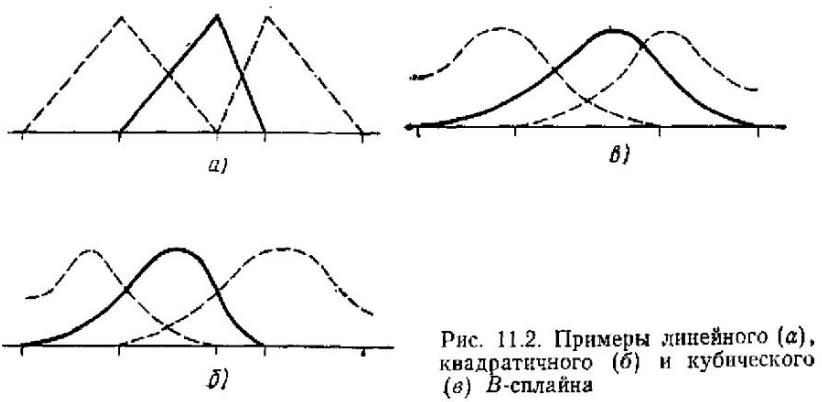


Рис. 11.2. Примеры линейного (*а*), квадратичного (*б*) и кубического (*в*) *B*-сплайна

Линейный:

$$N_{i,\,1}(x) = \begin{cases} \dfrac{x - x_i}{x_{i+1} - x_i} & \text{при } x_i \leqslant x \leqslant x_{i+1}, \\ \dfrac{x_{i+2} - x}{x_{i+2} - x_{i+1}} & \text{при } x_{i+1} \leqslant x \leqslant x_{i+2}. \end{cases} \qquad (11.6)$$

Квадратичный:

$$N_{i,2}(x) = \frac{(x - x_i)^2}{(x_{i+2} - x_i)(x_{i+1} - x_i)} \quad \text{при } x_i \leqslant x \leqslant x_{i+1}; \qquad (11.7а)$$

$$N_{i,\,2}(x) = \frac{(x - x_i)(x_{i+2} - x)}{(x_{i+2} - x_i)(x_{i+2} - x_{i+1})} +$$

$$+ \frac{(x_{i+3} - x)(x - x_{i+1})}{(x_{i+3} - x_{i+1})(x_{i+2} - x_{i+1})} \quad \text{при } x_{i+1} \leqslant x \leqslant x_{i+2}; \qquad (11.7б)$$

$$N_{i,\,2}(x) = \frac{(x_{i+3} - x)^2}{(x_{i+3} - x_{i+1})(x_{i+3} - x_{i+2})} \quad \text{при } x_{i+2} \leqslant x \leqslant x_{i+3}; \qquad (11.7в)$$

При равномерном размещении точек склеивания на сегментах с длиной L приведенные выражения существенно упрощаются. В

этом случае разумным допущением является $x_i = iL$ и удобно ввести нормированную переменную

$$u = (x - x_i)/L = x/L - i. \tag{11.8}$$

В результате уравнения, описывающие B-сплайны, принимают следующий вид. *Равномерно линейный:*

$$U_{i,1}((i+u)L) = \begin{cases} u & \text{при } 0 \leqslant u \leqslant 1, \\ 2-u & \text{при } 1 \leqslant u \leqslant 2. \end{cases} \tag{11.9}$$

Равномерно квадратичный:

$$U_{i,2}((i+u)L) = \frac{1}{2}u^2 \qquad \text{при } 0 \leqslant u \leqslant 1; \tag{11.10а}$$

$$U_{i,2}((i+u)L) = \frac{3}{4} - \left(u - \frac{3}{2}\right)^2 \quad \text{при } 1 \leqslant u \leqslant 2; \tag{11.10б}$$

$$U_{i,2}((i+u)L) = \frac{1}{2}(3-u)^2 \qquad \text{при } 2 \leqslant u \leqslant 3. \tag{11.10в}$$

В случае равномерного распределения точек склеивания уравнение (11.5) можно использовать непосредственно, что позволяет получить выражение для кубического B-сплайна по выражению для квадратического B-сплайна. *Равномерно кубический B-сплайн:*

$$U_{i,3}((i+u)L) = \frac{1}{6}u^3 \qquad \text{при } 0 \leqslant u \leqslant 1; \tag{11.11а}$$

$$U_{i,3}((i+u)L) = \frac{2}{3} - \frac{1}{2}(u-2)^3 - (u-2)^2 \quad \text{при } 1 \leqslant u \leqslant 2; \tag{11.11б}$$

$$U_{i,3}((i+u)L) = \frac{2}{3} + \frac{1}{2}(u-2)^3 - (u-2)^2 \quad \text{при } 2 \leqslant u \leqslant 3; \tag{11.11в}$$

$$U_{i,3}((i+u)L) = \frac{1}{6}(4-u)^2 \quad \text{при } 3 \leqslant u \leqslant 4. \tag{11.11г}$$

Используя B-сплайны в качестве базиса, произвольный сплайн можно представить в следующем виде:

$$p(x) = \sum_{i=-m}^{k-1} a_i N_{i,m}(x). \tag{11.12}$$

В этом уравнении имеется ровно $k + m$ параметров: $a_{-m}, a_{-m+1}, \ldots, \ldots, a_{k-1}$. Значение сплайна $p(x)$ на каждом подсегменте определяется суммой самое большее $m + 1$ B-сплайнов, так что сплайн $p(x)$, действительно, обладает свойством локальности. Изменение любого из коэффициентов уравнения (11.12) приводит к изменению вида кривой только на $m + 1$ сегментах. Следующий пример иллюстрирует некоторые характерные свойства как B-сплайнов, так и сплайнов в целом.

Пример 11.1. Требуется построить интерполяционный квадратичный сплайн по представляющим данные точкам (0,0), (1,1), (2,2), (3, *P*), (4,2), (5,1), (6,0). Влияние, оказываемое значением *P* на аппроксимацию, будет изучаться на примере равномерного B-сплайна при $L = 1$. Подстановка выражений из уравнения (11.10) в уравнение (11.12) и замена нормированной переменной u ее выражением (11.8) (при

$$p(x) = \frac{1}{2} a_i (x-i)^2 + a_{i-1} \left\{ \frac{3}{4} - \left[x - \left(i + \frac{1}{2} \right) \right]^2 \right\} +$$

$$+ \frac{1}{2} a_{i-2} (i+1-x)^2 \quad i \leqslant x \leqslant i+1; \qquad (11.13а)$$

выражение для производной на этом же сегменте имеет следующий вид:

$$p'(x) = a_i (x-i) - 2a_{i-1} \left[x - \left(i + \frac{1}{2} \right) \right] - a_{i-2} (i+1-x). \qquad (11.13б)$$

$L=1$) дает следующее

Оба уравнения справедливы при $i = 0, 1, ..., n-1$; их использование позволяет

$$p(i) = \frac{1}{2} [a_{i-1} + a_{i-2}], \qquad (11.14а)$$

$$p'(i) = a_{i-1} - a_{i-2}. \qquad (11.14б)$$

Для данной задачи интерполирования получаем набор уравнений

$$a_{-1} + a_{-2} = 1, \qquad a_0 + a_{-1} = 2,$$

$$a_1 + a_0 = 4, \qquad a_2 + a_1 = 2P,$$
$$a_3 + a_2 = 4, \qquad a_4 + a_3 = 2.$$
$$a_5 + a_4 = 0,$$

получить следующие выражения для точек склеивания:

Число неизвестных превышает число этих уравнений и для определенности можно задать касательную в одной из точек. Рассмотрим два случая.

Случай (а). Пусть $p(3)=0$, т. е. в средней из симметрично расположенных точек, представляющих данные, задается горизонтальная касательная. В результате $a_{-2} = a_5 = 2-P$, $a_{-1} = a_4 = -2 + P$, $a_0 = a_3 = 4-P$, $a_1 = a_2 = P$. Поскольку все коэффициенты зависят от значения P, изменение одной экспериментальной точки отражается на всем аппроксимационном сплайне. На рис 11.3 проиллюстрированы случаи: $P = 0$; 2; 2,5; 3.

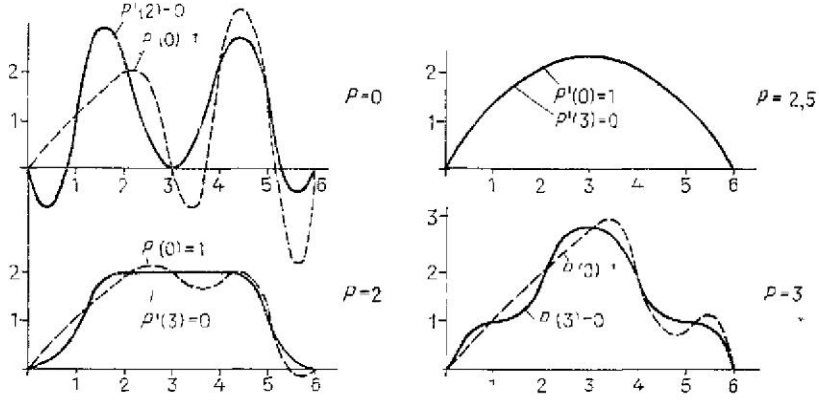


Рис. 11.3. Иллюстрация к примеру 11.1. Влияние положения экспериментальной точки (значения *P*) и задания касательной на форму квадратичного сплайна

Случай (б). Пусть $p'(0) = 1$. При этом интерполяционный сплайн является асимметричным и определяется коэффициентами $a_{-2} = -0{,}5$; $a_{-1} = 0{,}5$; $a_0 = 1{,}5$, $a_1 = 2{,}5$; $a_2 = 2P - 2{,}5$; $a_3 = 6{,}5 - 2P$; $a_4 = -4{,}5 + 2P$; $a_5 = 4{,}5 - 2P$.

Легко убедиться в том, что значения $p(x)$ на интервале $(0,2)$ равны значениям x, и, следовательно, значения сплайна на первых двух интервалах не зависят от значения P. Это достижение, однако, приводит к утрате симметрии. Графики $p(x)$ при четырех значениях P приведены на рис. 11.3.

Следует подчеркнуть, что окончательные результаты, т. е. значения $p(x)$, не зависят от выбора представления сплайна и мы получили бы то же самое, использовав, например, уравнение (11.2). Пример 11.1 и рис. 11.3 иллюстрируют несколько свойств сплайнов. Хотя сплайн-интерполяция обладает свойством локальности, построение, проводимое с учетом дополнительных степеней свободы, может оказать существенное воздействие на окончательный результат. Это обстоятельство, в сущности, не противоречит утверждению 11.1, которое сводится лишь к тому, что с помощью дополнительных точек склеивания кривую можно изменять локально. Читатель может в качестве упражнения попробовать в приведенном примере ввести на интервале (2, 4) вместо одной точки склеивания две (см. задачу 11.4).

Второй вывод заключается в том, что сплайны между экспериментальными точками могут претерпевать колебания столь же резкие, как и те, что свойственны интерполяционным многочленам. Причиной хороших результатов, полученных в разд. 10.2 посредством кусочной аппроксимации, является определенная свобода выбора точек склеивания, предоставившаяся в этом случае. В приведенном примере при использовании дополнительных точек склеивания результаты могли бы быть лучшими. Несложные вычисления позволяют убедиться в том, что значения сплайна 2,3 и 3,7 при $p'(x) = 1$ и $p'(x) = -1$ соответственно и $P = 3$ обеспечивают линейную аппроксимацию на интервалах (0,2) и (4,6). Этот же результат обеспечивается при значениях сплайна 2,5 и 3,5 и при $P = 0$. (В данном случае правильному выбору соответствуют следующие значения $p'(x)$: $p'(2) = 1$ и $p'(4) = -1$.)

Закончим этот раздел рассмотрением одного интересного свойства B-сплайнов.

Теорема 11.1. При произвольных значениях x и m

$$\sum_i N_{i,m}(x) = 1. \tag{11.15}$$

Замечание. Пределы суммирования не указаны в явном виде, поскольку сумма, соответствующая каждому значению x, содержит только $m+1$ членов.

Доказательство. Воспользуемся рекуррентным уравнением (11.5). Его можно переписать для $i-1$ вместо i

$$N_{i-1,m}(x) = \frac{x - x_{i-1}}{x_{i+m-1} - x_{i-1}} N_{i-1,m-1}(x) + \frac{x_{i+m} - x}{x_{i+m} - x_i} N_{i,m-1}(x). \tag{11.16}$$

Суммируя это уравнение с уравнением (11.5), получаем

$$N_{i-1,\,m}(x)+N_{i,\,m}(x)=\frac{x-x_{i-1}}{x_{i+m-1}-x_{i-1}}N_{i-1,\,m-1}(x)+$$

$$+N_{i,\,m-1}(x)+\frac{x_{i+m+1}-x}{x_{i+m+1}-x_{i+1}}N_{i+1,\,m-1}(x). \qquad (11.17)$$

Процесс получения уравнения (11.17) из уравнений (11.5) и (11.16) можно повторять для значений, скажем, от $i = j$ до $i = k$, что приведет в результате к следующему уравнению:

$$\sum_{i=j}^{i=k}N_{i,\,m}(x)=\frac{x-x_j}{x_{j+m}-x_{j+1}}N_{j,\,m-1}(x)+$$

$$+\sum_{i=j+1}^{i=k}N_{i,\,m-1}(x)+\frac{x_{k+m+1}-x}{x_{k+m+1}-x_{k+1}}N_{k+1,\,m-1}(x), \qquad (11.18)$$

Выбрав значение j достаточно малым, а значение k достаточно большим, можно обеспечить равенство нулю $N_{i,m-1}(x)$ и $N_{k+1,m-1}(x)$. В таком случае уравнение (11.18) можно записать в виде

$$\sum_{i}N_{i,\,m}(x)=\sum_{i}N_{i,\,m-1}(x). \qquad (11.19)$$

Другими словами, значение суммы B-спланов в точке x не зависит от значения m. Обратившись к уравнению (11.4), легко установить, что при $m = 0$ ее значение равно 1, на чем доказательство теоремы заканчивается.

11.4. ВЫЧИСЛИТЕЛЬНЫЕ АСПЕКТЫ ИСПОЛЬЗОВАНИЯ B-СПЛАЙНОВ

Уравнение (11.5) представляет простую вычислительную процедуру, обеспечивающую определение значения B-сплайна в точке x. Отметим, что любому сегменту $[x_i, x_{i+1}]$ соответствует лишь $m+1$ ненулевых B-сплайнов степени m. На этом сегменте значение $N_{i,m}(x)$ зависит лишь от значения $N_{i,m-1}(x)$, так как $N_{i+1,m-1}(x)$ на этом сегменте равно нулю, а $N_{i-l,m}(x)$ $(0<l<=m)$ зависит как от значения $N_{i-l+1,m-1}(x)$, так и $N_{i-l,m-1}(x)$. Эта взаимозависимость B-сплайна проиллюстриро-

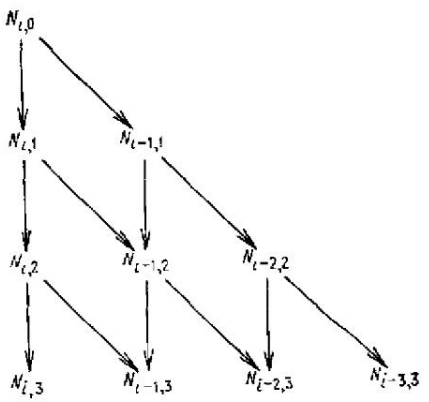


Рис. 11.4. Взаимозависимость значений B-сплайнов в точке x.

Каждый член сплайна представляет собой взвешенную сумму одного или двух членов, расположенных на верхней по отношению к нему строке Стрелки указывают направление вычислительного процесса Вертикальные линии обозначают умножение на первый множитель, а диагонали — умножение на второй множитель в уравнении (11.5)

вана на рис. 11.4. Для того чтобы определить значения В-сплайнов *m-й* степени, необходимо пройти *m*—1 предшествующих уровней «схемы, представленной на этом рисунке, и на каждом *из* них определить значения В-сплайнов от $N_{i,j}(x)$ до $N_{i-l,j}(x)$, где *j* — степень В-сплайна на соответствующем уровне и *l* пробегает значения от 0 до *j*. Описанная вычислительная процедура реализуется в виде алгоритма 11.1а.

Алгоритм 11.1 а. Процедура *BSPLINE (i, x, m).* Вычисление всех *B*-сплайнов в точке *x,* принадлежащей сегменту $[x_i, x_{i+i}]$.

Обозначения. m — степень сплайна, *x* — точка, в которой вычисляются значения сплайнов. Массив *N(I, J)* содержит значения $N_{I,J}(x)$. *a* и *b* — вспомогательные переменные.

1. *N (i,* 0) присваивается 1.
2. **For** *j* = 1 до *m* **do:**
   Begin.
3. **For** *i=0* **до** *j* **do:**
   Begin.
   {Вычисление *N (i—l, j).*}
4. $a=((x—x_{i-l})/(x_{i-l+j} — x_{i-l}))N(i — l, j—1).$
5. $b= ((x_{i-l+j+1}—x)/(x_{i-l+j+1} — x_{i-l+1}))N (i — l+l,j— 1).$
6. $N (i — l, j)=a + b.$
   End.
   End.
7. **Конец алгоритма.**

Число операций умножения и деления, выполняемых этим алгоритмом для каждого значения *x,* пропорционально m^2. Если точки склеивания расположены равномерно и для некоторого класса задач их положения фиксированы, то значения В-сплайнов можно определять непосредственно по заданным в явном виде формулам (11-9) — (11.11) или аналогичным им. Вычислительная сложность этого метода также пропорциональна m^2, но используемые операции проще.

11.5. ИНТЕРПОЛЯЦИОННЫЕ В-СПЛАЙНЫ

Пусть (t_1,y_1), $(t_2, y_2), \ldots,$ (t_n, y_n) — экспериментальные точки, для которых требуется построить интерполяционный сплайн. Известны способы решения этой задачи. Один из них предусматривает отождествление каждой экспериментальной точки с одним из узлов сплайна. Поскольку сплайн обладает $k + m$ степенями свободы, то при малых *m* (что обычно имеет место) можно принять $k = n—1$ и отождествлять точки склеивания с $t_1, t_2, \ldots, t_n$. В случае линейного сплайна (m=1) число степеней свободы равно ровно *n* и искомая кривая полностью определяется множеством прямых, соединяющих точки (t_i, y_i) с точками (t_{i+1}, y_{i+1}), где t=1, 2,..., ..., *n*—1. При $m=3$ общее число степеней свободы равно $n + 2$, в

связи с чем после введения ограничений, обеспечивающих прохождение искомой кривой через экспериментальные точки, две степени свободы остаются неиспользованными. При решении прикладных задач они обычно используются для задания касательных в концевых точках.

Другой подход предусматривает чередование точек склеивания с экспериментальными точками и, может быть, выполнение условия $k + m = n$. Рассмотрим этот случай подробно, причем для определения коэффициентов интерполяционного сплайна будем применять B-сплайны. Пусть, как это было введено в разд. 11.1, число точек склеивания равно k—1. В таком случае для $x_i <= t_j <= x_{i+1}$ должно иметь место следующее уравнение [см. (10.12)]:

$$a_i N_{i,m}(t_j) + a_{i-1} N_{i-1,m}(t_j) + \ldots + a_{i-m} N_{i-m,m}(t_j) = y_j,$$
$$1 \leqslant j \leqslant n. \qquad (11\,20)$$

Всего имеется n таких уравнений с $k + m$ неизвестными. Каждое из уравнений содержит т + 1 членов и, следовательно, соответствующая матрица разделена на слои, определяемые, по меньшей мере, m нижними и m верхними диагоналями. [Ср. с примером 11.1.] Точное описание кривой зависит от взаимного расположения точек склеивания и экспериментальных точек. Если точки склеивания расположены равномерно и, кроме того, совпадают с ординатами экспериментальных точек, то кривая имеет достаточно простую форму. Каждая точка задается значениями лишь m ненулевых B-сплайнов, которые определяются уравнениями (11.10) или (11.11). В случае квадратичного сплайна уравнения для параметров имеют вид

$$a_i + a_{i-1} = 2y_i, \quad i = 1, 2, \ldots, n, \qquad (11.21а)$$

в случае кубического сплайна

$$a_i + 4a_{i-1} + a_{i-2} = 6y_i, \quad i = 1, 2, \quad , n. \qquad (11\,21б)$$

К этой системе уравнений следует добавить еще ограничения, налагаемые на концевые точки. В противном случае их решение тривиально. Если предполагается, что искомая кривая — периодическая, то вместо наложения дополнительных ограничений на концевые точки уменьшается число неизвестных, для чего a_0 приравнивается a_n и т. д.

Общий случай оказывается несколько сложнее. Совершенно очевидно, что на каждом сегменте число экспериментальных точек не должно превышать т+1, так как в противном случае система может оказаться переопределенной. Если экспериментальные точки и точки склеивания перемежаются, то при этом может быть всего лишь $m+1$ ненулевых диагоналей. Важно выдерживать правильное соотношение между точками склеивания и экспериментальными точками, которое определяется следующей теоремой.

Теорема 11.2. Задача отыскания интерполяционного сплайна имеет единственное решение в том и только том случае, если

$$N_{jm}(t_j) \neq 0, \quad j = 1, 2, \ldots, n. \qquad (11.22)$$

Мы опустим доказательство этой теоремы (см. работу [11.15]) и остановимся лишь на ее интерпретации. На каждом подынтервале между точками склеивания имеется лишь $m + 1$ ненулевых B-сплайнов, и, следовательно, чтобы условия теоремы выполнялись, такой подынтервал не может включать больше $m+1$ экспериментальных точек. Это гарантирует против переопределенности системы уравнений (11.20). Известно также, что B-сплайн N_{jm} не равен нулю лишь на подсегменте (x_j, x_{j+m+1}) и, следовательно, это единственный подсегмент, который может содержать экспериментальную точку t_j. Поскольку число B-сплайнов в уравнении (11.12) равно числу экспериментальных точек $(n=k+m)$, возникают дополнительные ограничения. Первый из B-сплайнов является ненулевым только на первом подсегменте и, следовательно, этот подсегмент должен содержать какую-либо экспериментальную точку. То же самое относится к последнему подынтервалу. Если первый подынтервал содержит две экспериментальные точки, второй может вообще не содержать ни одной. В противном случае он должен содержать, по меньшей мере, одну экспериментальную точку и т. д.

Алгоритм 11.1 проверяет выполнение условий теоремы 11.2, задает систему уравнений вида (11.20) и затем решает ее, используя соответствующую библиотечную программу. На шаге 0 в концевых точках вводятся кратные узлы. Это необходимо для того, чтобы на каждом сегменте, содержащем экспериментальную точку, иметь $m + 1$ B-сплайнов. Кратность точек x_0 и x_k можно всегда задавать равной $m + 1$, не прибегая к проверкам. Некоторые из индексов массивов, используемых на шаге 4 алгоритма, могут оказаться отрицательными. Если программа, реализующая данный алгоритм, записывается на таком машинном языке, который не допускает использования отрицательных индексов, то алгоритм можно видоизменить, предусмотрев увеличение значений нижних индексов массива на m

Алгоритм 11.1. Интерполирование с помощью B-сплайнов

Обозначения, (t_j, y_j), $1<=j<=n$ - множество экспериментальных точек. x_i, $1<=i<=k+1$ — множество точек склеивания. m — степень сплайна, r — вспомогательная переменная. $B(j, I)$ — матрица системы уравнений (11.20): $B(j, I) = N_{I,m}(t_j)$.

0. If существует положительная вспомогательная переменная r такая, что $x_{m-r-1}<=t_1<=x_{m-r}$, **then** точка x_0 выбирается в качестве точки склеивания кратности $m—r+1$. **If** существует вспомогательная переменная r такая, что $x_{k-m+r}<=t_n<=x_{k-m+r+1}$, then точка x_k выбирается в качестве точки склеивания кратности $r+ 2$.
1. For j=1 до n проводится проверка условия $x_{j-m}<t_j<x_j$.
2. For i=1 до k проверяется, не превышает ли число экспериментальных точек $m+1$ на интервале (x_{i-1}, x_i).
3. **For** j=1 до n **do:**

Begin.

4 Отыскивается такое наибольшее i, при котором $x_i <= t_j$.

{Это означает, что $t_j <= x_{i+1}$ } For $I=i$ до i—m с шагом —1 **do Begin**

6 Производится обращение к процедуре *BSPLINE (i, t_j, m)*

7 *B (j, I)* приравнивается $N_{Im}(t_j)$

End

End

8 Используется одна из стандартных процедур решения линейных уравнений с расслоенными матрицами для определения $a_{_m}, a_{-m+1}, \ldots, a_{k-1}$ по *B(j, I)* и $y_1, y_2, \ldots, y_n$

9 Конец алгоритма.

Пример 11.2. Для экспериментальных точек (1 1), (31) (42) и (60) требуется построить интерполяционный сплайн первой степени на сегменте [0,7] с точками склеивания $x = 2$ и $x = 5$ В этом случае обе концевые точки должны быть двойными Нетрудно убедиться в том что интерполяционный сплайн первой степени состоит из прямой, соединяющей точки (31) и (42) (на сегменте [2,5]), прямой, соединяющей точки (1,1) и (20) (на сегменте [02]), и прямой, соединяющей точки (53) и (60) (на сегменте [57]) (рис 115) Читателю предлагается формальное обоснование этого результата найти самостоятельно в качестве упражнения (см задачу 11.6)

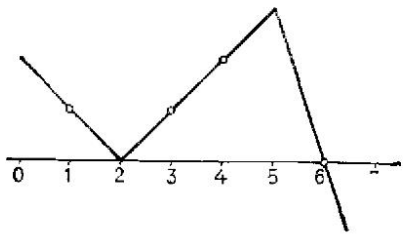


Рис 11.5 Линейный интерполяционный сплайн проходящий через четыре экспериментальные точки и имеющий две точки склеивания Взаимное расположение экспериментальных точек и точек склеивания иллюстрируют утверждение теоремы 11.2

11.6. ИРИМЕНЕНИЕ *B*-СИЛАЙНОВ В МАШИННОЙ ГРАФИКЕ

В-сплайны можно использовать для порождения кривых аналогично тому, как это делалось с помощью многочленов Безье Если P_i (i = 0, 1, , k) — множество точек ориентиров, то сплайн можно определить следующим образом

$$\mathbf{P}(t) = \sum_{i=0}^{k} \mathbf{P}_i N_{i,m}(t) \qquad (11\ 23)$$

Диапазон значений t больше не должен ограничиваться сегментом [0,1] и точки склеивания $t_1, t_2, \ldots, t_{k-1}$ должны задаваться несколько иначе, чем это было в случае многочленов Безье Отметим, что уравнение (11.23) эквивалентно двум парам уравнений

(11.12), причем точки ориентиры соответствуют коэффициентам этого уравнения a_i Очевидно, что P(t) представляет собой сумму векторов, умноженную на числа (5-сплайны), сумма которых в свою очередь в силу теоремы 11.1 равна единице Следовательно, имеет место следующий результат

Утверждение 11.2. Сплайн *P(t)* лежит в выпуклой оболочке самое большее $m+1$ точек-ориентиров Pi.

При *m=1* соответствующий многоугольник определяется точками ориентирами (Каждое значение сплайна *P(t)* лежит на прямой, соединяющей две точки) На рис 11.6,*а* приведены два примера для случая $m = 3$ — заштрихованными треугольниками обозначена та часть плоскости, в которой располагается кривая P(t)-

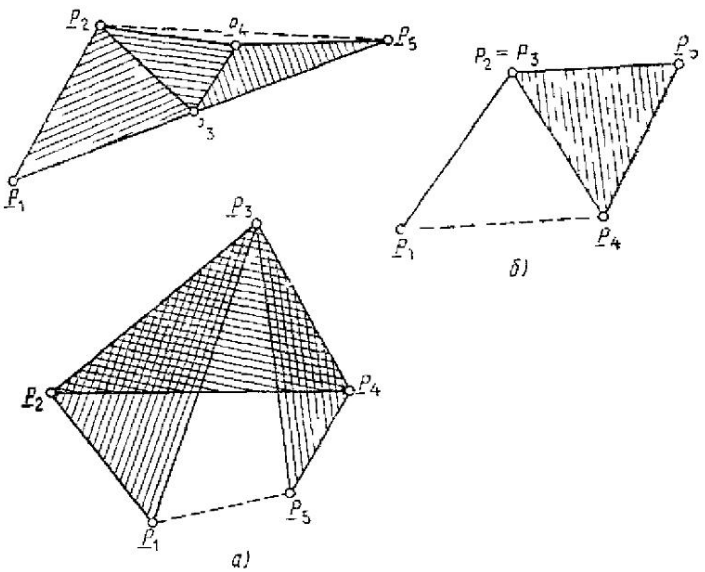


Рис. 11.6. Иллюстрация к утверждению 11.2

Заштрихованы те части плоскости, в которых должен располагаться квадратичный **сплайн с** заданными точками ориентирами Штриховые линии ограничивают часть плоскости, в которой должен располагаться многочлен Безье с теми же точками ориентирами

Наличие кратных точек вынуждает синтезируемую кривую проходить ближе к точкам-ориентирам, во многом аналогично тому, как это происходит в случае многочленов Безье На рис 11.6,*б* показано, что при $m = 2$ наличие двойных точек обеспечивает прохождение кривой через них Можно было бы наложить более жесткие ограничения на расположение сплайна (см. пример 11.4), однако сравнение заштрихованных областей на рис. И б и выпуклой оболочки всех точек (штриховые линии) показывает, что 5-сплайны обеспечивают значительно более точное управление формой воспроизводимой кривой, чем многочлены Безье.

Преимущество уравнения (11.23) состоит в том, что сплайн P может представляться трехмерным вектором, что обеспечивает получение таким способом пространственных кривых.

Можно предложить еще одну интерпретацию уравнения (11.23), если рассматривать P(t) и Pi как комплексные числа. В таком случае это просто комплексный сплайн

$$z(t) = \sum_{i=0}^{k} z_i N_{i,m}(t). \qquad (11.23')$$

Эти две разновидности задания сплайна эквивалентны, поскольку комплексные числа допускают геометрическую интерпретацию (координата x соответствует действительной части, а координата y — мнимой). Представление в комплексной форме широко использовалось Кнутом при синтезе шрифта (см. разд. 11.8).

Остановимся теперь более подробно на свойствах сплайнов, представляемых с помощью B-сплайнов. Пусть $B_{i,m,:i}(t)$—значения $N_{i,m}(t)$ при $t_{i+j}<=t<=t_{i+j+1}$; имеет смысл рассматривать значения $j = 0,1,\ldots,m$. В таком случае уравнение (11.23) можно переписать в следующем виде:

$$\mathbf{P}(t) = \sum_{j=0}^{m} \mathbf{P}_{i-j} B_{i-j,m,j}(t), \quad t_i \leqslant t \leqslant t_{i+1}. \qquad (11.24)$$

Если имеется лишь $k+1$ точек-ориентиров P_0, P_1 ..., P_k, то значения соответствующего сплайна можно определять лишь для $t>=t_m$. В общем случае $P(t_m)<>P_0$ и $P(1)<>P_k$. Для того чтобы гарантировать прохождение сплайна через концевые точки, последние должны быть кратными. Если кратность точки равна т, то соответствующий сплайн при любых обстоятельствах проходит через нее. Выше это было показано геометрически, кроме того, это следует и из уравнения (11.24). Если $P_{i-1} = P_{i-2} = \ldots = P_{i-m}$, то

$$\mathbf{P}(t) = \mathbf{P}_{i-1}\left[\sum_{j=1}^{m} B_{i-j,m,j}(t)\right] + \mathbf{P}_i B_{i,m,0}(t), \quad t_i \leqslant t \leqslant t_{i+1} \qquad (11.25)$$

В силу теоремы 11.1 член этого уравнения, заключенный в квадратные скобки, равен $1-B_{i,m,0}$. Из уравнения (11.5), кроме того, следует, что значения $B_{i,m,0}(t)$ пропорциональны $(t-t_i)$. Обозначив через c соответствующий постоянный коэффициент пропорциональности, получаем

$$\mathbf{P}(t) = \mathbf{P}_{i-1}[1 - c(t-t_i)^m] + \mathbf{P}_i c(t-t_i)^m. \qquad (11.26)$$

Из этого уравнения следует, что при $t = t_i$, $P(t_i) = P_{i-1}$. Более того, при $t_i<=t<=t_{i+1}$ значения искомого сплайна попадают на прямую, соединяющую векторы P_{i-1} и P_i.

Для того чтобы лучше понять свойства сплайнов, целесообразно обратиться к более простым примерам. В случае равномерного расположения точек склеивания можно воспользоваться представлениями B-сплайнов с помощью уравнений (11.9) — (11.11) и нормированной переменной u. Выражение (11.8) предусматривает за-

висимость задания u от индекса подсегмента i, в связи с чем необходимо проявлять определенную осмотрительность, используя на различных сегментах различные способы задания B-сплайнов. Нетрудно также показать, что при замене в выражениях, определяющих $U_{i-J,m}$, переменной u на $u+J$ соответствующие формулы становятся совместимыми с выражениями, определяющими $U_{i,m}$. В результате уравнение (11.24) принимает вид

$$m=1:\ \mathbf{P}\,(L(i+u))=u\,\mathbf{P}_i+(1-u)\,\mathbf{P}_{i-1}\ ; \tag{11.27а}$$

$$m=2:\ \mathbf{P}\,(L\,(i+u))+\frac{1}{2}\,u^2\,\mathbf{P}_i+\left[\frac{3}{4}-\left[u-\frac{1}{2}\right]^2\right]\mathbf{P}_{i-1}+$$

$$+\frac{1}{2}\,(u-1)^2\,\mathbf{P}_{i-2}\ ; \tag{11.27б}$$

$$m=3:\ \mathbf{P}\,(L\,(i+u))+\frac{1}{6}\,u^3\,\mathbf{P}_i+\left[\frac{2}{3}-\frac{(u-1)^3}{2}-(u-1)^2\right]\mathbf{P}_{i-1}+$$

$$+\left[\frac{2}{3}+\frac{u^3}{2}-u^2\right]\mathbf{P}_{i-2}+\frac{1}{6}\,(1-u)^3\,\mathbf{P}_{i-3}. \tag{11.27в}$$

При записи всех трех уравнений предполагается, что 0<=u<=1. Уравнения (11.27) можно представить в виде сумм степеней переменной u следующим образом:

$$m=1:\ \mathbf{P}\,(L\,(i+u))=(\mathbf{P}_i-\mathbf{P}_{i-1})\,u+\mathbf{P}_{i-1}\ ; \tag{11.28а}$$

$$m=2:\ \mathbf{P}\,(L\,(i+u))=\left[\frac{\mathbf{P}_i+\mathbf{P}_{i-2}}{2}-\mathbf{P}_{i-1}\right]u^2+$$

$$+[\mathbf{P}_{i-1}-\mathbf{P}_{i-2}]\,u+\frac{1}{2}\,[\mathbf{P}_{i-1}+\mathbf{P}_{i-2}]\ ; \tag{11.28б}$$

$$m=3:\ \mathbf{P}\,(i+u))=\frac{1}{2}\left[\frac{1}{3}\,\mathbf{P}_i-\mathbf{P}_{i-1}+\mathbf{P}_{i-2}-\frac{1}{3}\,\mathbf{P}_{i-3}\right]u^3+$$

$$+\left[\frac{1}{2}\,\mathbf{P}_{i-1}-\mathbf{P}_{i-2}+\frac{1}{2}\,\mathbf{P}_{i-3}\right]u^2+\frac{1}{2}\,[\mathbf{P}_{i-1}-\mathbf{P}_{i-3}]\,u+$$

$$+\frac{1}{3}\left[\frac{1}{2}\,\mathbf{P}_{i-1}+2\,\mathbf{P}_{i-2}+\frac{1}{2}\,\mathbf{P}_{i-3}\right]. \tag{11.28в}$$

Уравнения (11.28) удобно использовать для исследования свойств сплайнов. В частности, с их помощью можно получить аналоги уравнения (11.26) для случая $m=2$ и $m=3$. [Уравнение (11.27а) является таким аналогом для случая $m=1$.] Итак,

$$m=2:\ \mathbf{P}\,(L\,(i+u))=\frac{1}{2}\,[\mathbf{P}_i-\mathbf{P}_{i-1}]\,u+\mathbf{P}_{i-1}\ ; \tag{11.29а}$$

$$m=3:\ \mathbf{P}\,(L\,(i+u))=\frac{1}{6}\,[\mathbf{P}_i-\mathbf{P}_{i-1}]\,u^3+\mathbf{P}_{i-1}. \tag{11.29б}$$

В этих уравнениях $\mathbf{P}(L,\ i)=\mathbf{P}_{i-1}$. С другой стороны,

$$\mathbf{P}\,((i+1)\,L)=\frac{1}{2}\,(\mathbf{P}_{i-1}+\mathbf{P}_i)\ \text{ при }\ m=2, \tag{11.30а}$$

$$\mathbf{P}\,((i+1)\,L)=\frac{5}{6}\,\mathbf{P}_{i-1}+\frac{1}{6}\,\mathbf{P}_i\ \text{ при }\ m=3. \tag{11.30б}$$

Если $m+1$ точек-ориентиров лежат на прямой, то из утверждения 11.1 следует, что значения искомого сплайна также будут лежать на этой прямой. Уравнение (11.24) указывает, что последнее будет иметь место при $t_i <= t <= t_{i+1}$.

Для того чтобы строить замкнутые кривые, достаточно задать $P_{-1} = P_m$, $P_{-2} = P_{m-1}$ и т. д., а кроме того, $t_{-1} = t_m$, $t_{-2} = t_{m-1}$ и т. д.

Пример 11.3. а. Рассмотрим девять точек, представленных на рис. 11.7; здесь (P_0, P_1, P_2, P_3), (P_3, P_4, P_5, P_6) и (P_6, P_7, P_8, P_0) — множества коллинеарных точек. По этим точкам требуется построить равномерно квадратичный сплайн. Для определения точек Q_0, Q_1, ..., Q_8, где $Q_i = P(iL)$ воспользуемся уравнением (11.30а). Задав в уравнении (11.27*б*) $u=1/2$, получаем

$$\mathbf{P}\left(\left(i+\frac{1}{2}\right)L\right) = \frac{1}{8}\mathbf{P}_{i-2} + \frac{3}{4}P_{i-1} + \frac{1}{8}\mathbf{P}_i. \qquad (11.31а)$$

или

$$\mathbf{P}\left(\left(i+\frac{1}{2}\right)L\right) = \frac{1}{2}\left[\frac{1}{4}\mathbf{P}_{i-2} + \frac{3}{4}\mathbf{P}_{i-1}\right] + \frac{1}{2}\left[\frac{3}{4}\mathbf{P}_{i-1} + \frac{1}{4}\mathbf{P}_i\right]. \qquad (11.31б)$$

Уравнение (11.31*б*) определяет графический метод отыскания точек R_3, R_6 и R_8 (см. рис. 11.7). Если эти точки коллинеарны, то, естественно, P((i+1/2)L) совпадает с P_{i-1}.

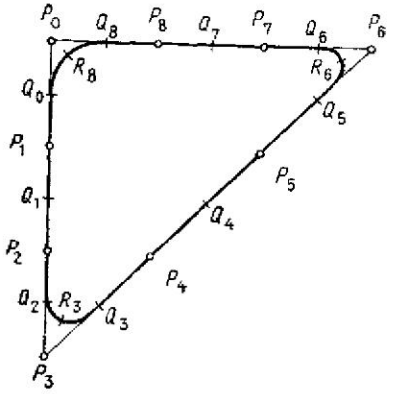


Рис. 11.7. Построение квадратичного сплайна, аппроксимирующего треугольник

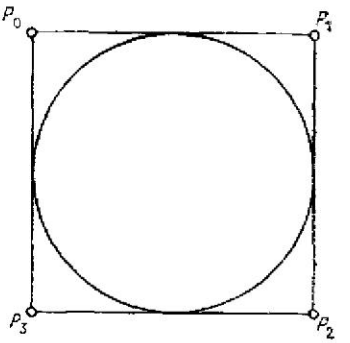


Рис. 11.8. Построение квадратичного сплайна, аппроксимирующего квадрат

б. На рис. 11.8 представлен периодический сплайн, построенный по четырем точкам, образующим квадрат. Хотя этот сплайн выглядит как окружность, он окружностью не является. Отношение длины радиуса этой «окружности», взятой по диагонали, к длинам ее радиусов, взятым по вертикали и горизонтали, составляет 3/(2 sqrt(2)) = 1,06.

Из уравнения (11.24) следует, что сплайны, построенные по точкам-ориентирам, в отличие от интерполяционных сплайнов локально зависят от самое большее m точек.

11.7. ОПИСАНИЕ ФОРМЫ ОБЪЕКТОВ И B-СПЛАЙНЫ

Встречаются прикладные задачи, в которых важно иметь очень точное описание формы кривой. Одним из примеров такой ситуации служит синтез очертаний символов для фотонаборных устройств, сопряженных с ЭВМ, который осуществляется математическими средствами. В некоторых выпускаемых промышленностью системах такого типа символы задаются с помощью вектора, содержащего код контура соответствующего символа. Полученный таким образом контур далее заполняется с помощью одного из алгоритмов типа, рассмотренного в разд. 8.2. В связи с тем, что этот метод основан на использовании векторов для представления линий, требуется задавать очень много таких векторов для того, чтобы очертания символов выглядели гладкими. (Наша книга напечатана с помощью такой системы.) Использование сплайнов для построения кривых по точкам открывает ряд интересных возможностей, поскольку в этом случае можно ограничиться заданием лишь точек-ориентиров, а не точек, действительно принадлежащих контуру. В следующем примере приводятся формулы, которые целесообразно использовать при решении указанных задач.

Пример 11.4. Рассмотрим форму квадратичного сплайна, точки склеивания которого распределены равномерно. Из уравнения (11.28*б*) следует, что значения сплайна $P(iL)$ всегда определяются уравнением (11.30а), т. е. соответствующий сплайн проходит через середины сторон многоугольника, образованного точками-ориентирами. Дифференцирование уравнения (11.27*б*) дает сле-

$$\frac{\partial}{\partial u}\mathbf{P} = [\mathbf{P}_i + \mathbf{P}_{i-2} - 2\mathbf{P}_{i-1}]\,u + [\mathbf{P}_{i-1} + \mathbf{P}_{i-2}]. \qquad (11.32)$$

Подстановка в уравнение (11.32) $u=0$ или $u=1$ показывает, что при при этих значениях нормированной переменной u соответствующий сплайн касается сторон многоугольника, построенного на точках-ориентирах Перегрушировка членов уравнения (11.27*б*) приводит к следующему выражению:

$$\mathbf{P}(u) = \frac{1}{2}[\mathbf{P}_i + \mathbf{P}_{i-1}]\,u^2 + 2(u - u^2)\,\mathbf{P}_{i-1} + \frac{1}{2}[\mathbf{P}_{i-1} + \mathbf{P}_{i-2}]\,(u-1)^2. \qquad (11.33)$$

дующий результат:

Поскольку, как можно убедиться, сумма $u^2+2(u-u^2) + (u-1)^2$ равна единице, то искомый сплайн лежит внутри треугольника, образованного вершиной многоугольника, построенного по точкам-ориентирам, и средними точками двух смежных с этой вершиной сторон.

Ограничения, налагаемые на расположения построенного в этом примере сплайна, являются значительно более жесткими, чем те, что были определены в утверждении 11.2. Таким образом, квадратичные сплайны с равномерным размещением узлов целесообразно использовать при решении таких прикладных задач, в которых требуется точное представление формы объекта и, в частности, необходимо воспроизвести гладкие очертания, опираясь на очень небольшое число экспериментальных точек. На рис. 11.9 представлены расположение исходных точек-ориентиров и построенный по ним сплайн, образующий очертания буквы. Изображение этой буквы определяется двенадцатью точками внешнего контура и во-

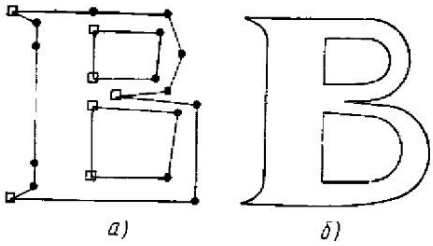


Рис. 11.9. Пример использования сплайнов, формируемых по точкам-ориентирам в базисе В-сплайнов для синтеза очертаний символов шрифта:
а — многоугольник, построенный по точкам-ориентирам, . простые точки-ориентиры, О точки-ориентиры, кратность которых равна двум; *б* — искомый сплайн

семью точками двух внутренних контуров (по четыре на каждый). В этом отношении квадратичные сплайны предпочтительнее кубических, для которых соотношение между многоугольником точек-ориентиров и искомой кривой является не столь жестким. Теоретически для аппроксимации некоторой заданной кривой с одной и той же степенью точности при использовании квадратичного сплайна требуется больше узлов, чем при использовании кубического. Однако при решении многих прикладных задач рост числа используемых узлов не имеет значения. Читатели могут убедиться в справедливости этого замечания, попытавшись построить очертания буквы *В,* приведенной на рис. 11.9 (или какого-либо аналогичного контура) с помощью кубического сплайна. Кубический сплайн удобнее использовать при решении таких прикладных задач, в которых форма объекта задается последовательностью точек и касательными в них к искомой кривой. В этих случаях дуги, соединяющие узлы, легко строятся с помощью выражения, аналогичного (10.5).

11.8. БИБЛИОГРАФИЧЕСКИЕ ЗАМЕЧАНИЯ

Сплайны рассматриваются, хотя бы кратко, в большинстве современных учебников по численным методам и более подробно — в большинстве книг, посвященных теории приближения. Во втором томе монографии Раиса [10.10] этой проблеме посвящена глава, причем основное внимание уделено теории сплайнов. Несколько книг посвящены исключительно сплайнам. В некоторых из них дается изложение общих положений математической теории сплайнов и не уделяется специального внимания задачам построения кривых по точкам [11.1, 11.12, 11.13], в других книгах основной упор делается как раз на эти задачи [11.14]. В сборниках [11.9, 11.10] помещены статьи, в которых содержится много материала по сплайнам. Де Бор много занимался изучением В-сплайнов [11.5, 11.6] и его монография [11.7] является наилучшим справочником по сплайнам для всех, сталкивающихся с их применением. В этой книге можно найти как результаты, относящиеся к теории сплайнов, так и тексты программ ЭВМ и конкретные примеры. В нашей книге разд. 11.4 и 11.5 основываются на работе [11.6]. В статье [11.3] рассматривается интересная разновидность сплайна.

Ризенфелд первый предпринял систематическое изучение использования В-сплайнов в машинной графике и указал на их пре-

имущества [11.11]. В сборнике [11.2] помещены статьи, посвященные использованию сплайнов в машинной графике и смежным проблемам. В статье [11.4] дан прекрасный обзор алгоритмов, обеспечивающих использование B-сплайнов для описания формы объектов. В частности, в ней приводится алгоритм построения сплайна, в котором используется разбиение, аналогичное рассмотренному в разд. 10.5 применительно к многочленам Безье. В статье приводится также обширный список литературы.

Кнут применил кубические сплайны для синтеза очертаний символов шрифта [11.8]. Он ввел понятие некоторого *меташрифта, в* котором каждый символ определяется кубическим сплайном с комплексными коэффициентами. Реальные символы порождаются как объединение круговых дисков или иных простых фигур, центры которых расположены в точках определяющих сплайнов.

11.9. ЗАДАЧИ

11.1. Является ли кривая, построенная при решении задачи 10.1, простым сплайном?

11.2. Рассмотрите следующий линейный сплайн, имеющий только одну точ ку склеивания:

$$p(t) = at - a(t - t_b)_+.$$

Вычислите его значение для $a=10^3$ и $f=10^{14}$ при $t_b=2$, ориентируясь на использование ЭВМ с 16-разрядными машинными словами и выполнение арифметических операций с одинарной точностью. Имеет ли значение, как задается сплайн: приведенным здесь выражением или уравнением (11.1)?

11.3. Требуется найти выражение или число, определяющие максимальные значения и значения в узлах для всех B-сплайнов, заданных уравнениями (11.6), (11.7), (11.9) —(11.11).

11.4. Рассмотрите пример 11.1 в случае ввода одной дополнительной точки склеивания на интервале (2,4).

11.5. Напишите программу, реализующую алгоритм 11.1. Будьте внима тельны при выборе стандартной программы решения системы уравнений с уче том того, что соответствующая матрица системы является расслоенной.

11.6. Проведите детальный анализ примера 11.2 и дайте строгое доказа тельство того, что кривая, приведенная на рис. 11.5, действительно представ ляет собой искомый интерполяционный сплайн.

11.7. Представьте уравнение (11.23') в полярных координатах. Можно ли посредством надлежащего выбора коэффициентов, точек склеивания и степеней B-сплайнов построить сплайн, представляющий собой окружность? Если Вы считаете, что это возможно, приведите в подтверждение своей точки зрения соответствующее формальное доказательство.

11.8. Требуется найти уравнение вида $f(x, y) = 0$, эквивалентное уравнению (11.28*б*). Используйте полученное выражение для доказательства того, что соот ветствующая кривая является параболой.

Указание: если кривая задается уравнением вида $ax^2 + 2bxy + cy^2 + dx + ey + f = 0$, то она является параболой при $b^2 = ac$.

11.9. Напишите программу, реализующую формулу (11.28*б*), и используйте ее для воспроизведения гладких кривых, задаваемых точками-ориентирами.

11.10. Включите программу, написание которой предусмотрено задачей 11.9, в программу интерактивного редактирования точек, написание которой было предусмотрено задачей 10.6, и сопоставьте затруднения, с которыми приходится сталкиваться при построении кривой заданной формы на основе использования многочленов Безье и с помощью B-сплайнов.

11.11. Воспользуйтесь ограничением из примера 11.4, предусматривающим принадлежность значений сплайна площади треугольника, для вывода простого

правила выбора интервала выборки, обеспечивающего воспроизведение изображений кривых с равномерным расположением точек.

11.12. Проанализируйте пример 11.4 для случая кубических сплайнов. В частности, воспользуйтесь уравнением (11.28в) для выполнения следующих заданий: а) определения расположения сплайна $P(iL)$; б) доказательства параллельности касательной к кривой в данной точке прямой, соединяющей точки P_{i-1} и P_{i-3}; в) вывода условия, определяющего такое расположение четырех точек-ориентиров, при котором точка перегиба искомой кривой не совпадает ни с одним из ее узлов.

11.13. На рис. 11.10 изображены четыре точки-ориентира одной из дуг кубического сплайна. Пусть точка A представляет значение сплайна P_{i-3}, точка B-сплайна P_{i-2}, точка C — сплайна P_{i-1} и точка D — сплайна P_i. Точки E и F делят сегмент AB на три равные части, точки G и Я делят таким же образом сегмент BC и точки J и K — сегмент CD. L — средняя точка сегмента FG и M — средняя точка сегмента HJ. Покажите, что дуга рассматриваемого сплайна обладает на сегменте $[iL, (i+1)L]$ следующими свойствами: а) начинается в точке L и является касательной к прямой FG в этой точке; б) заканчивается в точке Л! и является касательной к прямой HJ в этой точке; в) содержит точку перегиба, если век-горы FG и HJ образуют угол, больший 90^0.

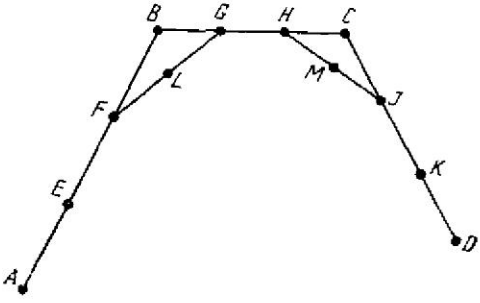


Рис. 11.10. Иллюстрация к задаче 11.13

11.14. Видоизмените уравнение (10.5) таким образом, чтобы его можно было использовать для построения кубического сплайна, проходящего через множество точек на плоскости и являющегося касательной к заданным прямым, проходящим через эти точки.

Глава 12 АППРОКСИМАЦИЯ КРИВЫХ

12.1. ВВЕДЕНИЕ

В последних двух главах речь шла в основном об интерполировании, т. е. о задачах, в которых было необходимо построить кривую, обязательно проходящую через все экспериментальные точки. (Точки-ориентиры относятся к параметрам синтеза, и поэтому кривую, задаваемую ими, также можно использовать для интерполирования.) Существует множество прикладных задач, при решении которых интерполирование не только не требуется, но даже нежелательно — в этих задачах требуется построить кривую, проходящую вблизи экспериментальных точек. Таким образом, возникает задача аппроксимации. Если построение кривой по точкам осуществляется в интерактивном режиме, то различие между двумя задачами несущественно. Пользователь просто изменяет параметры (такие, как точки-ориентиры) до тех пор, пока воспроизводимая кривая не принимает «правильный» вид. Последнее может означать, что соответствующая кривая проходит через все экспериментальные точки или через их большую часть, или вблизи всех экспериментальных точек и т. д. При построении кривой по точкам, осуществляемом в автоматическом режиме, такого рода субъективные критерии требуется заменить мерами близости, точно определенными математически. Наиболее распространенными из таких мер являются максимальная ошибка и суммарная квадратичная ошибка (СКО). Отклонение может измеряться либо по координате, либо по нормали к аппроксимирующей кривой. Последнее кажется более естественным, однако сложнее в вычислительном отношении. Пусть e_i обозначает ошибку в точке, измеренную в точке г, т. е. расстояние между построенной кривой и соответствующей точкой (измеренное любым из двух указанных методов). В таком случае

$$\text{максимальная ошибка} \equiv E_{\max} = \max_i |e_i| ; \tag{12.1а}$$

$$\text{суммарная квадратичная ошибка} \equiv E_2 = \sum e_i^2 . \tag{12.1б}$$

Поскольку математические средства СКО-аппроксимаций развиты в наибольшей степени, этот подход и является наиболее распространенным. Замкнутые формулы можно построить не только для случая аппроксимации с помощью многочленов, но и для ап-

проксимации с помощью сплайнов и иных кривых. В разд. 12.2 и 12.3 дается обзор важнейших свойств способов аппроксимации, предусматривающих использование в качестве аппроксимирующих кривых многочленов или сплайнов с фиксированными узлами. При определении аппроксимирующей кривой по критерию максимальной ошибки приходится решать задачу линейного программирования либо использовать какой-либо еще итеративный алгоритм. Таким образом, достижение оптимальной аппроксимации может потребовать значительного объема вычислений, в связи с чем наблюдается рост популярности субоптимальных эвристических алгоритмов. Наибольшую важность имеет задача, которая практически не поддается решению строго математическими средствами — речь идет об аппроксимации с помощью сплайна с переменными узлами. В разд. 12.4 и 12.5 обсуждаются некоторые аспекты этой задачи, однако рассмотрение ее в полном объеме выходит за пределы задач нашей книги (см. раз. 12.7). В разд. 12.6 представлены некоторые возможности использования методов построения кривых по точкам в машинной графике.

12.2. АППРОКСИМАЦИЯ ПО КРИТЕРИЮ СУММАРНОЙ КВАДРАТИЧЕСКОЙ ОШИБКИ

Определим аппроксимирующую кривую следующим образом:

$$g(t) = \sum_{j=0}^{m} a_j b_j(t), \tag{12.2}$$

где $b_j(t)$, $j=0, 1,..., m$ — семейство кривых, образующее базис. Говоря проще, это означает, что все представляющие интерес гладкие кривые можно некоторым единственным образом представить с помощью уравнения (12.2). Введение условия единственности приводит к следующему утверждению.

Утверждение 12.1. Пусть функции $b_0(t), ..., b_m(t)$ образуют базис. Если существуют коэффициенты $a_0,..., a_m$ такие, что

$$\sum_{j=0}^{m} a_j b_j(t) = 0 \text{ для всех } t, \tag{12.3}$$

то $a_0 = a_1 = ... a_m = 0$. Справедливо и обратное утверждение. Пусть F — множество, состоящее из m функций, которые удовлетворяют уравнению (12.3) лишь при равенстве всех коэффициентов a_j нулю. В таком случае, если другая функция может быть представлена в виде суммы функций $b_i(t)$, то это представление — единственно.

Доказательство. Допустим, что существуют такие ненулевые коэффициенты, для которых выполняется уравнение (12.3). Пусть

$$h(t) = \sum_{j=0}^{m} a'_j b_j(t), \tag{12.4}$$

гдё $h(t)$—произвольная функция. Суммируя уравнения (12.3) и (12.4), находим, что

$$h(t) = \sum_{j=0}^{m} (a_j + a'_j) b_j(t), \tag{12.5}$$

т. е. функцию $h(t)$ можно представить в виде суммы базисных функций более, чем одним способом, что однако, невозможно. Таким образом, утверждение доказано путем приведения к противоречию. Выполнение этих рассуждений в обратном порядке позволяет доказать обратное утверждение.

Рассмотрим теперь следующие заданные экспериментальные точки: (x_i, y_i), $i=0, 1, 2,..., n$. При изучении задачи аппроксимации параметр t удобно определять таким образом, чтобы его значения ставились в соответствие каждой экспериментальной точке, которые в результате представляются тройками вида (t_i, x_i, y_i). Это можно осуществлять двумя основными способами.

(а) *Сигналы* (зависимость сигнала от времени). Если последовательность значений координаты x_i является возрастающей, то

$t = (x_i - x_0)/x_n$ и обрабатываемые данные можно рассматривать как сигнал, заданный на сегменте [О, 1]. Тогда

$$e_i = y_i - g(t_i) = y_i - \sum_{j=0}^{m} a_j b_j(t_i), \tag{12.5а}$$

$$E_2 = \sum_{i=0}^{n} e_i^2 = \sum_{i=0}^{n} \left[y_i - \sum_{j=0}^{m} a_j b_j(t_i) \right]^2 . \tag{12.5б}$$

(б) *Контуры.* Другой способ определения параметра заключается в задании $t_i = iL$, где $L = 1/n$. В этом случае аппроксимация производится как по x, так и по y для значений t из сегмента [О, 1]:

$$e_i^x = x_i - g^x(t_i) = x_i - \sum_{j=0}^{m} a_j^x b_j(t_i), \tag{12.6а}$$

$$e_i^y = y_i - g^y(t_i) = y_i - \sum_{j=0}^{m} a_j^y b_j(t_i). \tag{12.6б}$$

Значения ошибки по координатам при этой можно минимизировать независимо:

$$E_2^x = \sum_{i=0}^{n} (e_i^x)^2 \quad \text{и} \quad E_2^y = \sum_{i=0}^{n} (e_i^y)^2 . \tag{12.7}$$

Поскольку уравнения (12.6) и (12.5а) аналогичны, мы рассмотрим лишь случай (а). Продифференцировав уравнение (12.5*б*) по a_k, получаем

$$\frac{\partial E_2}{\partial a_k} = -2 \sum_{i=0}^{n} \left[y_i - \sum_{j=0}^{m} a_j b_j(t_i) \right] b_k(t_i), \quad k = 0, 1, \ldots, m. \tag{12.8}$$

Минимизация значения E_2 предполагает равенство нулю всех правых частей последнего уравнения (соответствующих всем значениям k), откуда вытекает следующая система уравнений:

$$\sum_{j=0}^{m} \left[\sum_{i=0}^{n} b_k(t_i) b_j(t_i) \right] a_j = \sum_{i=0}^{n} b_k(t_i) y_i,$$
$$k = 0, 1, \ldots, m. \tag{12.9}$$

Можно показать, что для тех функций, для которых выполняется утверждение 12.1, система уравнений (12.9) имеет единственное решение и, по крайней мере в принципе, задача СКО-аппроксимации может быть сведена к решению уравнений (12.9). Задачу аппроксимации контура можно решить аналогичным образом. Заметим, что в последнем случае можно задать векторы

$$\mathbf{P}_i = \begin{bmatrix} a_i^x \\ a_i^y \end{bmatrix} \tag{12.10}$$

и определить точки-ориентиры уравнения (11.18).

На практике возникают трудности, главным образом, вследствие того, что матрица Грама

$$G_{kj} = \sum_{i=0}^{n} b_k(t_i)\, b_j(t_i) \quad k = 0, 1, \ldots, m; \quad j = 0, 1, \ldots, m \qquad (12.11)$$

для некоторых из чаще всего используемых базисных функций является почти особой. В частности, так обстоит дело при использовании в качестве базисных функций степеней t и m больше двух. Это свойство легко устанавливается при замене сумм интегралами, т. е. при введении допущения ->• . В таком случае для базисных функций — степеней t — матрица Грама имеет вид

$$G_{kj} = \int_0^1 t^k\, t^j\, dt = \int_0^1 t^{k+j}\, dt = \left[\frac{t^{k+j+1}}{k+j+1}\right]_0^1 = \frac{1}{k+j+1}. \qquad (12.12)$$

Значения соответствующих определителей равны 1/12 при m=1, 1/2160 при m = 2 и около (1/6) • 10^{-6} при $m=3$[1].

12.3. АППРОКСИМАЦИЯ С ПОМОЩЬЮ *B*-СПЛАЙНОВ

Ни один результат предыдущего раздела не зависит от особенностей строения функций $b_i(t)$, если они образуют базис. Рассмотрим теперь *B*-сплайны m-й степени, заданные на сегменте, содержащем k—1 точек склеивания. Общее число таких функций составит $k+m$—1. Допустим, что определено такое множество коэффициентов a_i что

$$\sum_{i=-m+1}^{k} a_i\, N_{i,m}(t) = 0 \text{ для всех } t. \qquad (12.13)$$

Это означает, что на любом сегменте сумма значений определенных *B*-сплайнов также будет равна нулю. При m=1 и равномерном расположении точек склеивания возникают уравнения вида

$$a_{i-1}(1-s) + a_i\, s = 0, \qquad (12.14)$$

где s= $(x—x_i)/L$. Очевидно, что уравнение (12.14) не может быть тождеством относительно s, за исключением случая, когда a_{i-1} и a_i равны нулю. Аналогичным образом можно проанализировать и общий случай ($m>1$ и точки склеивания расположены неравномерно), и, следовательно, можно показать, что *B*-сплайны действительно образуют *базис*. При этом в соответствующей матрице Грама ненулевые элементы будут расположены лишь на главной диагонали и m диагоналях, расположенных по обе стороны от главной. В самом деле, $N_{im}(t)$ и $N_{jm}(t)$ имеют нулевые значения на сегменте лишь при $|i-j|<=m$. Если снова в уравнении (12.11) перейти от сумм к интегралам, то для $m=1$ и равномерного разме-

[1] Малое значение определителя само по себе не служит индикатором сложности обращения матрицы Грама, однако обстоятельное обсуждение этой проблемы выходит за пределы задач нашей книги

щения точек склеивания находим, что первый и последний элементы главной диагонали равны $L/3$, элементы, расположенные на главной диагонали между ними, равны $2L/3$ и элементы диагоналей, расположенных выше и ниже главной, равны $L/6$. В случае двух сегментов определитель матрицы Грама равен $L^3/18$, трех сегментов — $5L^4/144$ и т. д. (см. задачу 12.1). Обращение таких матриц осуществляется много проще, чем обращение матрицы уравнения (12.12).

При решении прикладных задач важную роль играет вычисление значений элементов матрицы Грама при произвольном расположении точек склеивания в соответствии с уравнением (11.5). Это легко осуществить с помощью процедуры BSPLINE, представленной в виде алгоритма 11.1 а. Если доступный объем памяти позволяет это, то значения всех B-сплайнов могут храниться в ней до вычисления скалярных произведений.

Основной проблемой, возникающей при использовании B-сплайнов или иных кусочно-полиномиальных функций, является выбор расположения точек склеивания. К обсуждению этой проблемы мы и переходим.

12.4. АППРОКСИМАЦИЯ С ПОМОЩЬЮ СПЛАЙНОВ С ПЕРЕМЕННЫМИ ТОЧКАМИ СКЛЕИВАНИЯ

Работая с интерактивной системой, пользователь может воспроизвести экспериментальные точки, а затем с помощью программы-редактора точек типа рассмотренной в разд. 10.8 задать расположение точек склеивания. Существует, однако, много прикладных задач, при решении которых хотелось бы иметь возможность выбирать такие точки автоматически. К сожалению, последнее связано с решением очень трудной математической задачи. Дело здесь в следующем. Пусть T — вектор, представляющий расположение точек склеивания $t_1, \ldots, t_{l_1}$. Для того чтобы «проявить» зависимость базиса от точек склеивания, заменим в уравнениях функции $b_3(t)$ функциями $b_3(t, \mathbf{T})$. Так, уравнение (11.6) при помощи тривиальных изменений в обозначениях можно переписать в следующем виде:

$$b_j(t, \mathbf{T}) = \begin{cases} (t - t_j)(t_{j+1} - t_j), & t_j \leqslant t \leqslant t_{j+1}; \\ (t_{j+2} - t)/(t_{j+2} - t_{j+1}), & t_{j+1} \leqslant t \leqslant t_{j+2}. \end{cases} \tag{12.15}$$

Для получения оптимальной аппроксимации при нефиксированном расположении точек склеивания (положение точки склеивания является переменной) необходимо, кроме производных, задавае-

$$\frac{\partial E_2}{\partial t_k} = -2 \sum_{i=0}^{n} \left[y_i - \sum_{j=0}^{m} a_j b_j(t_i, \mathbf{T}) \right] \left[\sum_{j=0}^{m} \frac{\partial b_j}{\partial t_k} \right]. \tag{12.16}$$

Для примера, представленного уравнением (12.15), находим, что

$$\partial b_j(t, \mathbf{T})/\partial t_i = (t - t_{j+1})/(t_{j+1} - t_j)^2, \quad t_j \leqslant t \leqslant t_{j+1}. \tag{12.17}$$

мых уравнением (12.8), определить следующие:

Два последних уравнения показывают, что положения точек склеивания входят в оптимизационные уравнения нелинейно — в явном виде эта задача не решается. Поэтому следует обратиться к интерактивным методам, при использовании которых сначала произвольно выбирается множество положений точек склеивания, а затем эти положения начинают варьироваться с тем, чтобы уменьшить ошибку E_2.. Подобная процедура может потребовать много времени и по многим причинам является непрактичной.

В качестве еще одного примера возникающих затруднений рассмотрим уравнение (11.1), сформулированное без введения ограничений. Если в этом случае в уравнении (12.1*б*) заменить суммы интегралами, то ошибка задается следующим выражением:

$$E_2 = \sum_{k=0}^{r} \int_{t_k}^{t_{k+1}} e_k^2(t)\, dt, \qquad (12\ 18)$$

где ошибка в точке e_k определяется как

$$e_k(t) = y_i - \sum_{j=0}^{m} a_j^k b_j(t). \qquad (12\ 19)$$

Если выбран базис, не зависящий в явном виде от расположения точек склеивания (например, степени параметра *t*), то оказывается, что

$$\frac{\partial E_2}{\partial t_k} = e_{k-1}^2(t_k) - e_k^2(t_k). \qquad (12\ 20)$$

Здесь использован тот факт, что производная интеграла по его верхнему пределу интегрирования равна значению подынтегрального выражения в этой точке, а именно

$$\frac{d}{dx} \int_a^x f(t)\, dt = f(x)$$

Аналогичным образом производная интеграла по его нижнему пределу интегрирования равна отрицательному значению подынтегрального выражения в соответствующей точке. Уравнение (1220) имеет простую интерпретацию. при оптимальном положении точки склеивания абсолютные значения ошибок в точках одинаковы по обе стороны от нее Однако этим соотношением нельзя воспользоваться, так как значения этих ошибок зависят от положения точек склеивания Если же ввести их в уравнение (1220) в явном виде, то снова возникнет нелинейное уравнение

12.5. АППРОКСИМАЦИИ, ОСНОВАННЫЕ НА ИСПОЛЬЗОВАНИИ МНОГОУГОЛЬНИКОВ

Строго говоря, все кривые, воспроизводимые в большинстве систем обработки графической информации, представляют собой многоугольники, поскольку строятся они посредством соединения смежных пикселов линиями, для чего используется какая-либо

команда типа *vec(x,y)* (см. разд. 1.7, табл. 1.1). (Устройство обеспечивающее отображение только точек, представляет единственное существенное исключение.) Хорошо известный математический результат гласит, что всякую кривую можно аппроксимировать с произвольно заданной точностью некоторым многоугольником (ломаной) и, следовательно, соответствующая аппроксимация может выглядеть очень гладкой[1]. Если задано математическое описание кривой *C* вида

$$f(x, y) = 0, \qquad (12\,21)$$

то можно поставить задачу отыскания многоугольника, аппроксимирующего кривую *C с* высокой точностью и имеющего минимально возможное число вершин. Естественно, в случае, когда положение вершин можно изменять, удается находить более эффективные представления. Аналогичная задача возникает в распознавании образов с тем отличием, что С задается не уравнением, а множеством экспериментальных точек. Построение многоугольника по таким точкам способствует определению признаков, существенных для описания формы объекта Это проиллюстрировано на рис. 12.1: для зашумленных контуров с помощью алгоритма, описанного ниже, построены аппроксимирующие многоугольники.

Использование интерактивных графических устройств приносит мало пользы в обоих случаях. В первом случае для воспроизведения изображения кривой необходимо построить соответствующий многоугольник Во втором случае все операции должны выполняться автоматически В то же время мы уже столкнулись с тем, что задача оптимизации размещения узлов сплайна не имеет решения Для того чтобы получить хотя бы некоторое субоптимальное решение, не связанное с вычислительными проблемами, приходится упрощать как процедуру построения линий по точкам, так и процедуру размещения точек склеивания. Мы, в частности, не настаиваем на минимизации числа узлов, но лишь пытаемся довести его до значения, близкого к минимальному. Основная идея метода заключается в изучении групп точек и проверке их приближенной коллинеарности Если точки оказываются неколлинеарными,

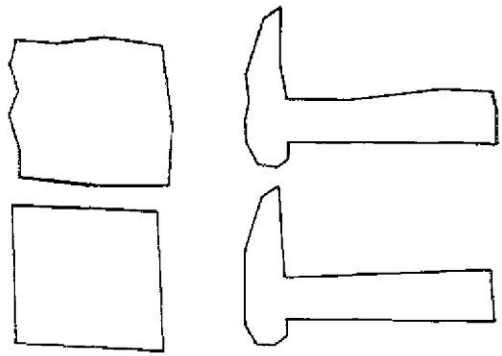


Рис 12 1 Примеры аппроксимации с помощью многоугольников, проведенной посредством алгоритма 12 1: *вверху* — исходные контуры рисунка, *внизу* — аппроксимирующие многоугольники

[1] Такой результат, естественно, сопряжен с определенными затратами, см. разд. 11.7 и рис. 11.9

заданная группа точек делится до тех пор, пока условие коллинеарности не оказывается выполненным. С другой стороны, группы объединяются, если в результате слияния возникает группа приближенно коллинеарных пикселов. Итак, используется процедура расщепления — слияния, результат которой обладает одним специфическим свойством, раскрываемым в следующем утверждении.

Утверждение 12.2. Число сегментов, получаемых в результате применения процедуры расщепления — слияния, всегда меньше удвоенного значения минимального числа таких сегментов.

Доказательство. Пусть $I_1, I_2, \ldots, I_m$ — сегменты минимального разбиения и $J_1, J_2, \ldots, J_n$ — сегменты, выделенные в результате применения процедуры расщепления — слияния. Ни один из сегментов I_i не может содержать двух сегментов J_k и J_{k+1} за исключением случая слияния последних. Любой сегмент I_i однако, может покрывать такой сегмент J_k, что сегменты J_{k-1} и J_{k+1} перекрываются с сегментами I_{i-1} и I_{i+1} соответственно (рис. 12.2). Таким образом, m сегментов могут полностью покрываться сегментами, принадлежащими оптимальному разбиению, и, кроме того, имеются еще сегменты, число которых равно числу точек разбиения, т. е. общее число сегментов равно m + m - 1. Следовательно, $n<=2m$— 1.

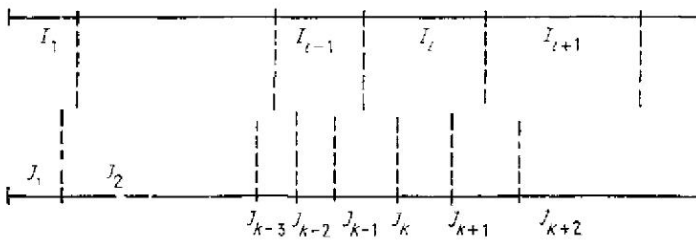


Рис. 12.2 Структура сегментов, используемая при доказательстве утверждения 12.2

Оказывается, что полученная оценка очень сильно завышена: в реальных условиях n принимает значения, много меньше $2m - 1$. Пример, дающий значение $n=2m$—1, предусматривает довольно вычурное взаимное расположение оптимальных точек склеивания и концевых точек групп пикселов, проверяемых на коллинеарность. Общий принцип организации алгоритма расщепления — слияния можно реализовывать различными способами и в подразд. 12.5.2 будет изложен один из простейших. (Сведения о более совершенных алгоритмах можно найти в разд. 12.7.)

12.5.1. СУБОПТИМАЛЬНЫЙ АЛГОРИТМ ПОСТРОЕНИЯ ЛИНИЙ ПО ТОЧКАМ

Любой алгоритм построения многоугольника по точкам предусматривает разбиение экспериментальных точек на группы, каждая из которых должна затем аппроксимироваться одной из сто-

рон многоугольника. Первое упрощение, которое можно ввести при решении задачи построения многоугольника по точкам, состоит в проведении прямой между концевыми точками каждой группы вместо поиска оптимальной аппроксимации. Прямая, используемая для аппроксимации точек (x_j, y_j), (x_{j+1}, y_{j+1}),..., (x_h, y_k), определяется уравнением

$$x(y_j - y_k) + y(x_k - x_j) + y_k x_j - y_j x_k = 0. \qquad (12.22)$$

В сущности, это уравнение (10.2), преобразованное посредством умножения на знаменатель и замены $p_2(x)$ на y, 1 на j и 2 на k. Приведем теперь один полезный результат.

Утверждение 12.3. Если точка (u, v) не принадлежит прямой, определяемой уравнением (12.22), то ее расстояние от этой пря-

$$d = u(y_j - y_k) + v(x_k - x_j) + y_k x_j - y_j x_k, \qquad (12.23)$$

$$L = \sqrt{(y_j - y_k)^2 + (x_k - x_j)^2}. \qquad (12.24)$$

Доказательство. Почленное деление уравнения (12.23) на уравнение (12.24) приводит к получению следующего выражения:

$$\frac{d}{L} = -u \sin\varphi + v \cos\varphi + c \qquad (12.25)$$

мой равно d/L, где

(угол φ определен на рис. 12.3,*а*). Константу c можно определить в явном виде из уравнений (12.23) и (12.24), однако лучше дать ей более простую интерпретацию. Если (X, Y) — произвольная точка, принадлежащая рассматриваемой прямой, то задав в уравнении (12.25) $u = X$ и $v = Y$, находим, что $d = 0$, поскольку точка (X, Y) должна удовлетворять уравнению (12.22). В таком случае эту константу можно задать как $c = -X\sin\varphi - Y\cos\varphi$ и переписать уравнение (12.25) следующим образом:

$$d/L = -(u - X)\sin\varphi + (v - Y)\cos\varphi. \qquad (12.26)$$

Обратимся теперь к чертежу, приведенному на рис. 12.3,*б*. Здесь m обозначает расстояние точки от рассматриваемой прямой. Очевидно, что $|BC| = (u - X)\sin\varphi$ и $|BD| = (v - Y)\cos\varphi$. В таком случае расстояние между точкой и прямой действительно равно

$$m = |CD| = |BD| - |BC| = (v - Y)\cos\varphi - (u - X)\sin\varphi$$

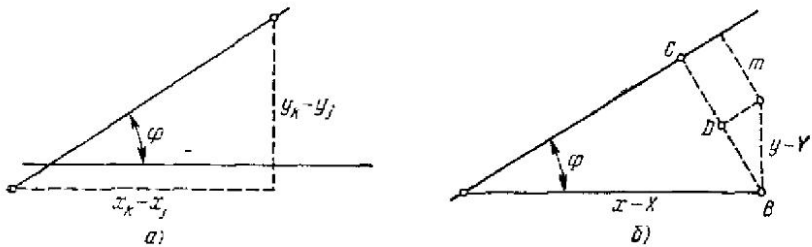


Рис. 12.3 Иллюстрация к утверждению 12.3 и его доказательству

Отметим, что знак при d/L указывает, с какой стороны от прямой находится точка.

Простую проверку коллинеарности обеспечивает вычисление левой части уравнения (12.25) для всех промежуточных точек. Можно задать максимальное значение расстояния таким образом, что точки множества не будут считаться коллинеарными, если какая-либо из них отстоит от прямой на расстояние, большее заданного максимума. С практической точки зрения этот прием не всегда оказывается целесообразным. В самом деле, ошибкам, обладающим регулярностью неслучайного характера, можно поставить в соответствие весовые коэффициенты. Допустимой мерой случайности служит число изменений знака величины d (рис. 12.4,*а*). Если вместо абсолютного максимального значения используется значение, нормированное по длине прямой L, то пороговое значение становится равным d/L^2.

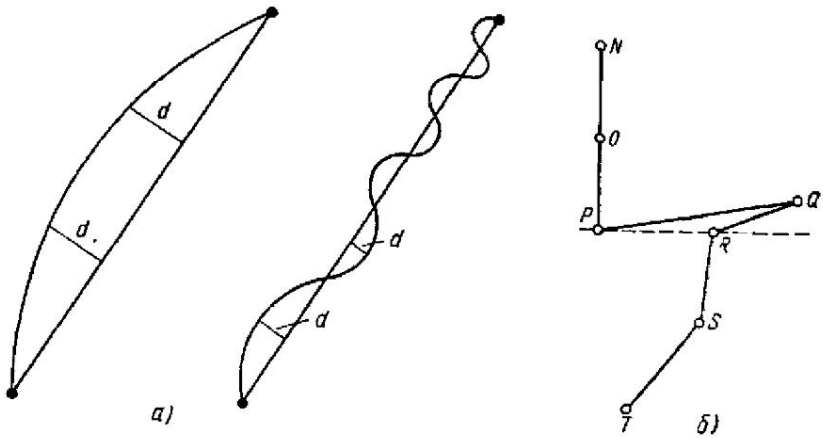


Рис. 12.4. Проверка коллинеарности точек:
а — ошибка d измеряется по нормали к прямой в каждом примере ошибка определена в двух точках, в первом знак ошибки не изменяется, во втором — изменяется шесть раз; *б* — особый случай, максимальная ошибка близка к нулю, однако при рассмотрении точек в качестве некоторого упорядоченного множества они оказываются неколлинеарными

Эти тесты на коллинеарность дают неверные результаты в особых случаях типа проиллюстрированного на рис. 12.4,*б*. На нем представлено упорядоченное множество точек P, Q и R, которое является частью большего контура, включающего ряд других точек, также изображенных на этом рисунке. Расстояние от точки Q до прямой, соединяющей точки P и R, очень мало, однако эти три точки не коллинеарны, если рассматривать их как упорядоченную последовательность. Для того чтобы обойти такие особые случаи, следует с длиной прямой L сравнивать длину дуги L_a, измеренную вдоль соответствующей последовательности точек. Если отношение L_a/L слишком велико, то рассматриваемые точки не могут быть коллинеарны. Это простой тест и его целесообразно ис-

пользовать до проведения остальных вычислений. Практический опыт работы с данными и вычислительные эксперименты с простыми геометрическими моделями свидетельствуют о том, что при $L_a/L<1,1$ допущение о коллинеарности можно принимать, не используя дополнительные тесты, а при $L_a/L>1,5$ коллинеарность исключается полностью.

Проведение прямой между крайними точками вместо поиска оптимальной аппроксимации не вызывает серьезных отклонений от оптимального решения, так как максимальное удаление точек от интерполяционной линии не более чем в двое превышает их удаление от оптимальной аппроксимирующей линии (см. задачу 12.3).

Эти положения реализованы в процедуре *COLLINEAR* (*j*, *k*, *m*, a, *L*), представленной в виде алгоритма 12.1. Шаг 1а ориентирован на особый случай и может быть опущен, если появление таковых не ожидается. На шаге 8 используются формулы для вычисления нормированной и ненормированной максимальной ошибки 7. Алогоритм можно упростить, исключив из него шаги 6, 7 и все остальные операции с d_0 и C. В таком случае проверка, осуществляемая на шаге 9, будет сведена к сравнению T (максимального расстояния от точки до прямой) и T_0. Лучшие результаты, однако, дает использование теста, определенного на плоскости T—C (рис. 12.5). Формальное обоснование данного подхода можно получить, рассматривая проверку коллинеарности как проверку некоторой гипотезы, однако обсуждение этой интерпретации выходит за пределы задач нашей книги и мы предлагаем использовать область допустимых значений, приведенную на рис. 12.5, в качестве некоторой эвристики.

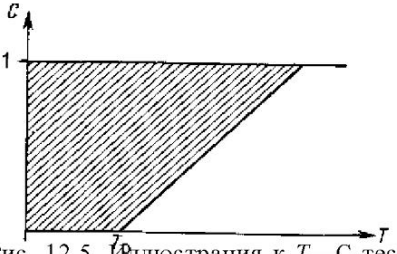


Рис. 12.5. Иллюстрация к T—С-тесту коллинеарности. Область допустимых значений числа изменений знака ошибки C и ошибки T заштрихована

Опыт практического использования подобных алгоритмов свидетельствует о том, что критерии, приводящие к получению наилучших результатов в одной прикладной задаче, при переходе к другой могут потерять это качество. Если, например, основным источником ошибок служит квантование, то использование максимальной ненормированной ошибки дает лучшие результаты, чем при нормировании по длине. Если, с другой стороны, приходится строить прямые по экспериментальным точкам, полученным посредством дискретизации рисунков, сделанных от руки, то лучшие результаты достигаются при использовании нормирования. Обоснованием использования нормирования в этом случае является то обстоятельство, что длинные линии человек обычно вырисовывает менее тщательно, чем короткие.

Читателю следует иметь в виду, что алгоритм 12.1 должен реализовываться на основе арифметических операций с плавающей

запятой, а не целочисленной арифметики. Неплохо, вероятно, кроме того, заранее разделить значения $a(1)$, $a(2)$ и $a(3)$ на L вместо того, чтобы в конце делить значение ошибки T на L^2 или L.

Алгоритм 12.1. Процедура *COLLINEAR* (j, k, m, a, L).

Обозначения. *Массивы* $x(\,)$ и $y(\,)$ определены как общие массивы, к которым процедура имеет доступ; j — индекс первой, k — индекс последней точек группы, проверяемой на колинеарность. Размерность массива a равна трем; он содержит коэффициенты прямой, соединяющей точку $x(j)$, $y(j)$ с точкой $x(k)$, $y(k)$. m — индекс точки, которой соответствует максимальное (по абсолютной величине) значение ошибки, т. е. точка $x(m)$, $y(m)$ находится на наибольшем расстоянии от прямой, соединяющей первую и последнюю точки. L — длина этой прямой. В результате выполнения процедуры определяются m, массив a и L. Используются следующие локальные переменные: T — максимальная ошибка (абсолютная величина), d — ошибка в некоторой точке, C — число изменений знака ошибки.

1. Вычисляются: $a(1)=y(j)-y(k)$,
$$a(2)=x(k)-x(j),$$
$$a(3)=y(k)x(j)-y(j)x(k),$$
$$L=\sqrt{a(1)^2+a(2)^2}.$$

1а. {**Факультативный шаг**}. Вычисляется общая **длина дуги**

$$L_a=\sum_{i=j+1}^{k}\sqrt{[x(i)-x(i-1)]^2+[y(i)-y(i-1)]^2}.$$

If отношение L_a/L больше 1,5, **then return** *ложь*.
If это отношение меньше 1,1, **then return** истина.

2. Задаются $m=j$, $T=0$, $d_0=0$ и $C=0$.
3. **For** $i=j+1$ до $k-1$ **do**:
Begin.
4. Вычисляется $d=a(1)x(i)+a(2)y(i)+a(3)$.
5. **If** $|d|$ больше T, **then** принимается $T=|d|$ и $m=i$.
6. **If** знак d отличается от знака d_0, **then** C увеличивается. {Если d_0 имеет нулевое значение, то C не изменяется.}
7. Задается $d_0=d$.
End.
8. Ошибка T заменяется T/L^2 (или T/L), и, если $k>j+2$, то C заменяется $C/(k-j-2)$. {Отметим, что если $k=j+2$, то $C=0$}.
9. **If** точка (C, T) расположена в заштрихованной части плоскости (рис. 12.5), **then return** *истина*.
10 **Else return** *ложь*.
11. **Конец алгоритма.**

12.5.2. ПРОСТОЙ АЛГОРИТМ ПОСТРОЕНИЯ МНОГОУГОЛЬНИКА ПО ТОЧКАМ

Алгоритм 12.2 обеспечивает построение многоугольника по точкам, обрабатывая небольшие группы точек размером k_0, так что $k=j+k_0$. (Типичные значения k_0 лежат в диапазоне 5—10). Алгоритм строит (отслеживает) текущую прямую L\ и новую прямую L_2. Если при применении соответствующего теста обнаруживается коллинеарность группы точек (проверка на шаге 4 устанавливает, что в результате использования процедуры *COLLINEAR* получено *истинное* значение), то выполняется блок шагов 5—10. На шаге 5 определяется новая прямая L_2, представляющая собой прямую, полученную в результате применения процедуры *COLLINEAR*. Если первая точка группы назначается точкой склеивания ($p=j$), то новая прямая становится текущей прямой (шаг 6), а шаги 7—9 пропускаются. Если p отличается от j, то на шаге 8 используется тест на слияние, заключающийся в сравнении прямых L_1, и L_2 (рис. 12.6).

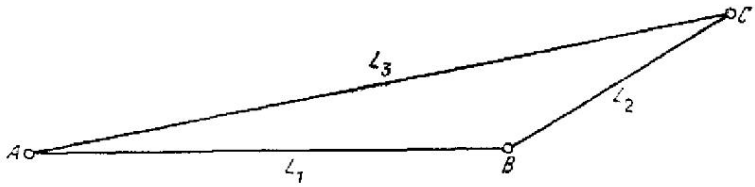


Рис. 126. Иллюстрация к алгоритму 12.2: задание прямых L_1, L_2 и L_3

Если угол между этими двумя прямыми мал, то соответствующие группы точек объединяются и текущая прямая определяется как прямая, соединяющая точки $x(p)$, $y(p)$ и $x(k)$, $y(k)$ (шаг8), Малый угол гарантирует близость новой аппроксимирующей прямой L_3 к обеим прямым L_1 и L_2 (см. утверждение 12.4). Если угол между прямыми L_1 и L_2 велик, то на шаге 9 алгоритм назначает точку $x(j)$, $y(j)$ точкой склеивания, задавая $p=j$. В любом случае алгоритм переходит к обработке новой группы точек, присваивая k значение j и увеличивая k на k_0.

Если тест на коллинеарность, проводимый на шагах 3 и 4, приносит отрицательный результат, то он применяется к группе точек, расположенных между первой точкой и точкой, в которой ошибка максимальна (по абсолютной величине). Для этого используется шаг 11, на котором задается $k = m$ и осуществляется возврат к началу цикла, что обеспечивает повторное выполнение шага 3 (Отметим, что при выполнении цикла приращение j не происходит автоматически. В цикле лишь проверяется, не дошел ли алгоритм до обработки последней экспериментальной точки.)

Операции, предусмотренные на шагах 8 и 9, можно заменить повторным применением теста на коллинеарность к объединению

сегментов, которые обеспечиваются обращением к процедуре *COLLINEAR (p, k, m,* a, *L*). Этот прием увеличивает общий объем вычислений, однако его использование является непосредственной гарантией адекватности теста.

Алгоритм 12.2 относится к описанному в литературе классу алгоритмов, осуществляющих «просмотр вдоль» некоторого множества точек. В каждой точке прямая корректируется и проверяется, не превышает ли ошибка допустимого значения. Если значение ошибки находится в допустимых пределах, алгоритм переходит к обработке следующей экспериментальной точки, в противном случае вводится соответствующая точка склеивания (см. разд. 12.7). Представленный ниже алгоритм относится к типу алгоритмов, осуществляющих «просмотр скачками вдоль» некоторого множества точек. Этот алгоритм предусматривает использование одной прямой для группы точек, что позволяет исключить вычисления, связанные с пересчетом прямой в каждой точке. Однако при возникновении ошибки алгоритм должен вернуться к точке, в которой ошибка принимает максимальное значение. Таким образом, важно выбирать размер группы точек k_0 достаточно малым для снижения частоты возвратов. При k_0=1 этот алгоритм сводится к алгоритму «просмотра вдоль». В этом случае тест на коллинеарность всегда адекватен и алгоритм сводится к шагам 8 и 9.

Алгоритм 12.2. Построение многоугольника по точкам

Обозначения. Массивы *x()* и *y()* содержат координаты экспериментальных точек. *j* — индекс первой, а *k* — индекс последней точки группы, проверяемой на коллинеарность. k_0 — число экспериментальных точек, обычно входящих в состав такой группы *p* — индекс последней вершины многоугольника и L_1— прямая, соединяющая точку *x(p), y(p)* с точкой *x(1), y(j). m* — индекс точки, в которой при проведении текста на коллинеарность была зафиксирована максимальная ошибка.

1. Задается $p = 0, j = 0$ и $k = k_0$.
2. **While** *j* меньше *n* — 1 do:
   Begin.
3. Обращение к процедуре *COLLINEAR* (*j, k, m,* a, *L*).
4. If результат, выдаваемый процедурой, имеет значение *истина,* then **do:**
   Begin.
5. Задается прямая L_2, для чего используются коэффициенты a, определенные с помощью процедуры *COLLINEAR.*
6. If *p* равно *j*, **then** задается прямая L_1 посредством приравнивая ее прямой L_2.
7. **Else do:**
   Begin.
8. If углом между прямыми L_1 и L_2 мал, **then** задается прямая L_1 как прямая L_3, соединяющая точки *x(p), y(p)* и *x(k), y(k).*

9. Else p приравнивается j и задается прямая L_1 посредством приравнивания ее прямой L_2.
End.

10. Значение j приравнивается k, и k увеличивается на k_0.
End.

11. Else задается $k = m$.
End.

12. Конец алгоритма.

12.5.3. СВОЙСТВА АЛГОРИТМА 12.2

Суммируем свойства, которыми обладает аппроксимация, осуществляемая на основе допущений на отклонения, использованные в процедуре *COLLINEAR* и на шаге 8 алгоритма 12.2.

Утверждение 12.4. Если на шаге 9 процедуры *COLLINEAR* ненормированное расстояние до прямой сравнивается лишь с фиксированным пороговым значением T_0 и если угол ω между двумя прямыми (шаг 8 алгоритма 12.2) меньше, чем $\sin^{-1}(T_0/L)$, где L — максимальная длина прямых L_2 и L_3, то максимальное расстояние до прямой L_3 меньше, чем $2T_0$.

Доказательство. В основе доказательства лежит ряд тригонометрических соотношений. Объекты, используемые при доказательстве, определены на рис. 12.7. Отметим, что с прямыми L_1 и

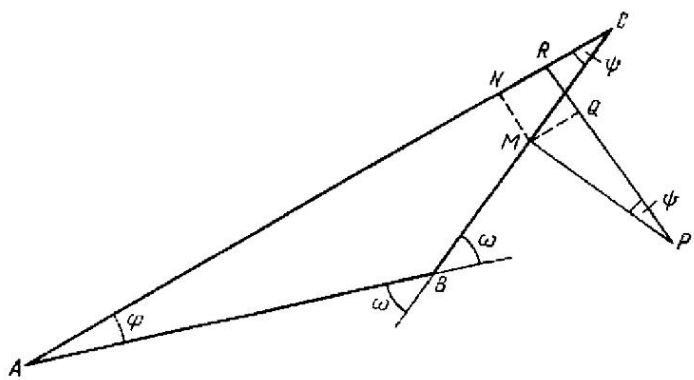


Рис. 12.7. Иллюстрация к доказательству утверждения 12.4

L_2 можно оперировать совершенно одинаково, т. е. в качестве прямой L_2 можно использовать как AB, так и BC и т. д., в качестве L_3 используется AC. Поскольку новая ошибка определяется отрезком $|PR|$, то

$$|PR| = |PQ| + |QR| = |PQ| + |MN| = |MP| \cos\psi + |CM| \sin\psi.$$

Поскольку ABC — треугольник, угол ψ меньше или равен углу

ω (ω = ψ + φ. Равенство углов ψ и ω наступает в случае, когда они оба равны нулю. Следовательно,

sin sin ψ <= *sin* ω <= T_0/L.

Поскольку cos^^1, получаем

/PR/<=/MP/ + /CM/ T_0/L.

По определению |MP|<=T_0 Кроме того, |CM|<=L. Следовательно, $|PR/<=T_0+L(T_0/L)=2T_0$.

Оценка, вводимая утверждением 12.4, несколько завышена из-за усиления неравенств в процессе доказательства. Хотя нами было показано, что алгоритм 12.2 обеспечивает построение аппроксимирующего многоугольника, расположенного вблизи от аппроксимируемых точек (и также непрерывного), мы не показали, что число сегментов, входящих в аппроксимацию, близко к минимальному. Если допустить, что экспериментальные точки получены в результате квантования некоторой выпуклой кривой, то неравенства, используемые в доказательстве утверждения 12.4, становятся менее свободными и удается показать, что число вершин построенного многоугольника близко к минимальному. Однако, поскольку при решении прикладных задач редко удается иметь дело лишь с выпуклыми кривыми, здесь не будем приводить это доказательство.

12.6. ПРИМЕНЕНИЕ МЕТОДОВ АППРОКСИМАЦИИ КРИВЫХ В МАШИННОЙ ГРАФИКЕ

Приложения методов аппроксимации кривых с помощью многоугольников или сплайнов с переменными узлами слишком многообразны, чтобы можно было исчерпывающе представить их в данной книге Краткий обзор некоторых из таких приложений проиллюстрирует важность этого направления для обработки графической информации Одним из возможных способов использования программы-редактора точек, описанной в разд. 10.8, является ввод в нее набросков для обеспечения воспроизведения соответствующих изображений в «очищенном» виде. Например, пользователь может указать на кривую, сообщив системе, что она должна представлять собой окружность, в результате чего указанная кривая будет заменена аппроксимирующей окружностью. Для выполнения таких заданий программа-редактор должна быть расширена: во-первых, должна быть предусмотрена возможность указывать группу точек, и, во-вторых, в нес должны быть введены процедуры аппроксимации функциями.

12.6.1. ОБРАБОТКА ГРУПП ТОЧЕК С ПОМОЩЬЮ ПРОГРАММЫ-РЕДАКТОРА ТОЧЕК

При этом виде обработки лучше всего использовать иерархическую структуру данных Определим структуру *кривая* следующим образом:

структура *кривая* – {*p, t, s*},

где *p* — указатель, определяющий структуру *точка,* содержащий первую точку дуги, *t* — тип структуры и *s* — переменная состояния. Тип определяет, каким образом следует интерпретировать точки кривой: как точки-ориентиры многочлена Безье, сплайна или экспериментальные точки, аппроксимирующие один из набора базовых объектов — окружность, квадрат, условное изображение резистора, условное изображение усилителя и т. д. Состояние указывает, подвергался ли объект стиранию. Можно также обобщить структуру *точка* (см. разд. 10.8.1) таким образом, чтобы в нее входил указатель, определяющий структуру *кривая,* к которой принадлежит каждая точка, что, однако, требуется не всегда. В сущности, процедура *place (к, у, q),* описанная в разд. 10.8.2, должна быть организована следующим образом.

Алгоритм 12.3. Процедура *place (x, y, q, Q)*

Обозначения, x и *y* — заданные координаты, *q* — указатель, определяющий *изаточку,* и *Q* — указатель, определяющий структуру *кривая,* выдаваемую процедурой.

1. For всех указателей *Z*, указывающих кривую с ненулевым *со стоянием,* do:
   Begin.
2. Задается *z* как Z->p.
3. Отыскивается пара *z -> x, z -> y,* являющаяся ближай шей к точке *(x, y); d* — расстояние между этими точ ками. (Просмотр последовательности точек осущест вляется посредством замены *z* на *z -> n;* просмотр прекращается при *z,* равном 0.)
4. If Z указывает первую кривую, задается Q = Z, *q = z* и *D = d.* **Else** проводится сравнение *d* и *D* и, если пер вое оказывается меньшим, Q заменяется на Z, *q* — на *z* и *D* на *d.*
   End.
5. **Конец алгоритма.**

Если команде, снабженной адресом, предшествует слово *объект* (или соответствующая аббревиатура «о»), то соответствующая операция применяется ко всей кривой. Указателем, определяющим кривую, служит значение Q, получаемое на выходе процедуры типа приведенной выше. В частности, первая точка кривой-указывается следующим образом:

(Q -> p) + *x, (Q -> p) -> y.*

12.6.2. ПОСТРОЕНИЕ ПРОСТЫХ АППРОКСИМИРУЮЩИХ КРИВЫХ

Рассмотрим несколько примеров объектов, форма которых может быть описана математически.

Пример 12.1. Отыскание наилучшего приближения множества точек окружностью по критерию минимума суммарной квадратичной ошибки эквивалентно

выбору x_c, y_c и r таким образом, чтобы была обеспечена минимизация величины

$$E = \sum_{i=1}^{n} |(x_i - x_c)^2 + (y_i - y_c)^2 - r^2|. \qquad (12.27)$$

(Предполагается, что ошибка в точке определяется по нормали к кривой, а не по осям координат.) Минимизация ошибки E — нелинейная задача; поэтому можно попытаться найти субоптимальное решение на основе выбора точки с координатами x_c и y_c в качестве центра тяжести экспериментальных точек. (Этим приемом можно пользоваться лишь при равномерном расположении экспериментальных точек по окружности.) В таком случае значение E можно минимизировать по r. Наилучшим является выбор r, равным радиусу инерции (см. задачу 12.5). На рис. 12.8 приведен пример использования такой субоптимальной схемы решения. (Экспериментальные точки показаны зачерненными кружочками.)

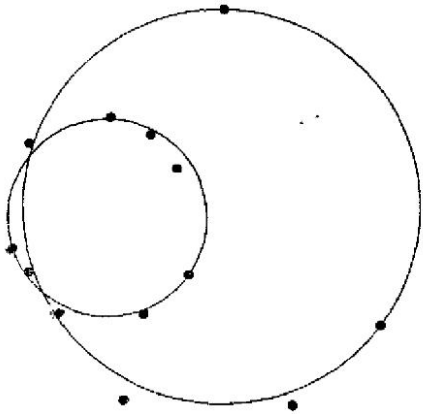


Рис. 12.8. Два множества точек, аппроксимированные окружностями

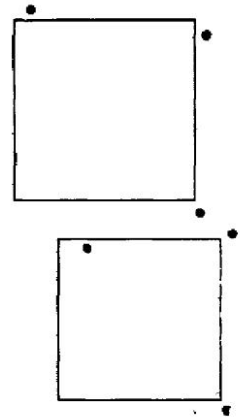


Рис. 12.9. Два множества точек, аппроксимированные квадратами

Пример 12.2. Заданы четыре точки, якобы образующие правильный квадрат, хотя на самом деле это не так. Необходимо изменить положение каждой точки таким образом, чтобы они образовали квадрат и сумма всех перемещений точек была минимальна. Следует выбрать значения x, y и a таким образом, чтобы была обеспечена минимизация следующего выражения:

$$E = (x_1 - x)^2 + (y_1 - y)^2 + (x_2 - x)^2 + (y_2 - y - a)^2 + \\ + (x_3 - x - a)^2 + (y_3 - y - a)^2 + (x_4 - x - a)^2 + (y_4 - y)^2. \qquad (12.28)$$

Взяв соответствующие частные производные, устанавливаем, что значения x, y и a должны удовлетворять следующей системе уравнений:

$$4x + 2a = \sum_{i=1}^{4} x_i,$$

$$4y + 2a = \sum_{i=1}^{4} y_i,$$

$$2x + 2y + 4a = y_2 + x_3 + y_3 + x_4.$$

Решение этой системы уравнений имеет вид:

$$a = \frac{1}{4}\,[y_2 + y_3 - y_1 - y_4 + x_3 + x_4 - x_1 - x_2], \tag{12.29а}$$

$$x = \frac{3}{8}\,(x_1 + x_2) + \frac{1}{8}\,(x_3 + x_4) - \frac{1}{8}\,(y_2 + y_3) + \frac{1}{8}\,(y_1 + y_4), \tag{12.29б}$$

$$y = \frac{3}{8}\,(y_1 + y_4) + \frac{1}{8}\,(y_2 + y_3) - \frac{1}{8}\,(x_3 + x_4) + \frac{1}{8}\,(x_1 + x_2). \tag{12.29в}$$

На рис. 12.9 даны иллюстрации использования этих формул. (Экспериментальные точки показаны зачерненными кружочками.)

Пример 12.3. Требуется по заданному множеству из n точек построить эллипс. Пусть a и b — длины осей эллипса и φ — угол, образованный одной из них с осью x. Выберем снова центр тяжести (x_c, y_c) в качестве центра эллипса. Определим моменты инерции относительно этого центра:

$$M_{xx} = \sum_{i=1}^{n} (x_i - x_c)^2, \quad M_{yy} = \sum_{i=1}^{n} (y_i - y_c)^2,$$

$$M_{xy} = \sum_{i=1}^{n} (x_i - x_c)\,(y_i - y_c)\ .$$

Можно показать, что при выборе угла φ, равного половине угла, тангенс которого равен $M_{xy}/(M_{yy} - M_{xx})$, моменты инерции M^c_{xx} и M^c_{yy} относительно осей эллипса будут максимальны по одной оси и минимальны по другой. (Момент инерции M^c_{xy} будет равен 0.) Наша цель состоит в том, чтобы уравнение эллипса было справедливо в среднем, т. е.

$$\sum_{i}^{n} \left(\frac{x_i^2}{a^2} + \frac{y_i^2}{b^2} - 1 \right) = 0, \tag{12.30}$$

где координаты x_i и y_i определяются в новой системе координат, начало которой совпадает с центром эллипса, а координатные оси параллельны осям

$$\frac{1}{a^2}\,M^c_{xx} + \frac{1}{b^2}\,M^c_{yy} - n = 0.$$

Это уравнение выполняется, если

$$a = \sqrt{M^c_{xx}/pn}, \quad b = \sqrt{M^c_{yy}/qn},$$

эллипса. Уравнение (12.30) эквивалентно уравнению

где $p+q=1$.

При выборе $p=q=1/2$ характер распределения (форма расположения) точек сохраняется: эллипс, длина осей которого равна указанным величинам, а экспериментальные точки равномерно расположены вдоль его контура, имеет моменты относительно обеих осей, равные (с точностью до масштабного коэффициента) исходным моментам.

12.7. БИБЛИОГРАФИЧЕСКИЕ ЗАМЕЧАНИЯ

Аппроксимация с помощью многочленов — это стандартная тема для большинства книг по численным методам анализа, в частности таких, как [10.5, 10.6, 10.10]. Материал, посвященный аппроксимации с помощью сплайнов с фиксированными узлами, мо-

жно найти в монографии [11.7], сборнике [11.9] и др. Хотя сплайны с переменными узлами и являются одним из лучших средств аппроксимации экспериментальных точек гладкой кривой, применять их нелегко, а литературы по этой проблеме довольно немного. В статье [12.2] отыскиваются некоторые важнейшие свойства таких сплайнов. Метод получения субоптимального решения представлен в статье [12.1]. Алгоритм решения одной частной задачи аппроксимации изложен в статье [12.5]. Эта работа интересна не только в связи с представленным алгоритмом, но также и статистическими данными, характеризующими вычисления в зависимости от степени многочлена, числа узлов и ошибки аппроксимации. Аппроксимации, полученные с помощью многоугольников с переменными положением и числом вершин, оказываются проще; они обсуждаются в монографии [3.7] и статьях [12.3, 12.4]. Эти работы содержат подробные описания алгоритмов расщепления — слияния. В статье [12.4] этот метод используется для построения кривых по точкам, а в статье [12.3] — для задания начальных условий в методе Ньютона. В монографии [3.7] описываются эвристические методы, в том числе метод просмотра вдоль множества точек, предложенных различными авторами. Наибольшее сходство с идеей, положенной в основу алгоритма 12.2, обнаруживает метод, предложенный Томеком [12.7], и метод многоугольников с минимальным периметром [12.6].

12.8. ЗАДАЧИ

12.1. Проверьте значения элементов и определителя матрицы Грама, приведенные в разд. 12.3. Покажите, например, что

$$U_{i,1}(x)\,U_{i,1}(x) = \int_{iL}^{iL+L}\left(\frac{x-iL}{L}\right)^2 dx + \int_{iL+L}^{iL+2L}\left(\frac{iL+2L-x}{L}\right)^2 dx = \frac{2L}{3}$$

и т. д.

12.2. Напишите программу, реализующую алгоритм 12.2.

12.3. Докажите утверждение, содержащееся в подразд. 12.5.1 и характеризующее соотношение максимальной ошибки интерполяционного многоугольника и оптимального аппроксимирующего многоугольника. (Если в процессе доказательства у Вас возникнут затруднения, обратитесь к статье [11.6].)

12.4. Усовершенствуйте программу-редактор точек, написанную в соответствии с задачей 10.6, таким образом, чтобы появилась возможность адресации дуг, и используйте ее для построения многоугольников по точкам, включив в программу-редактор программу, написанную в соответствии с задачей 12.2.

12.5. Докажите, что построение субоптимальной окружности требует выполнения следующего условия:

$$r = \frac{1}{\sqrt{n}}\sqrt{\sum_{i=1}^{n}(x_i-x_c)^2 + \sum_{i=1}^{n}(y_i-y_c)^2}\,.$$

12.6. Выведите уравнения (12.29) посредством прямой минимизации величины E с помощью уравнения (12.28). Используется ли некоторое допущение относительно упорядоченности экспериментальных точек? Измените алгоритм таким образом, чтобы его результаты были независимы от порядка, заданного на точках. Обобщите его и на случай аппроксимации с помощью прямоугольников.

12.7. Программу-редактор точек можно обобщить таким образом, чтобы она имитировала чертежника посредством ряда стандартных программ аппроксимации и с ее помощью можно было бы превращать небрежные наброски в аккуратные чертежи. Обеспечьте эту возможность посредством включения алгоритма 12.2 в программу-редактор. (Данная задача является продолжением аадачи 12.4.)

12.8. Напишите программу, реализующую формулы, приведенные в приме ре 12.3, и включите ее в программу-редактор с тем, чтобы она обеспечивала построение эллипсов, аппроксимирующих грубые наброски кривых, представляющих эллипсы.

Глава 13

ПРИБЛИЖЕНИЕ И ВОСПРОИЗВЕДЕНИЕ ПОВЕРХНОСТЕЙ

13.1. ВВЕДЕНИЕ

Воспроизведение *поверхности* трехмерного объекта играет важную роль при использовании систем обработки графической информации для решения таких прикладных задач, в которых пользователю необходимо располагать различными видами группы твердых тел. Приближение поверхности требуется осуществлять и в случаях, когда объекты задаются множествами точек. Приближение и воспроизведение поверхностей представляет интерес и для других прикладных областей. В машинной картографии и географии работа ведется с моделями участков местности, которые представляются математическими описаниями поверхностей. В обработке изображений и распознавании образов поверхности используются при анализе изображений, по крайней мере, двумя способами. Во-первых, изображение можно рассматривать как поверхность, используя значения яркости в качестве третьей координаты. Во-вторых, можно рассматривать поверхности объектов, присутствующих на изображении. В этом случае задача сегментации (см. гл. 4) сводится к идентификации групп пикселов, являющихся изображениями поверхности.

Поверхности, встречающиеся во всех этих областях, слишком сложны для того, чтобы их можно было описать единственным уравнением во всей их области определения, и, следовательно, возникает потребность в кусочной аппроксимации поверхностей. Простейшими являются кусочно-линейные аппроксимации, в частности построенные посредством использования многогранника с треугольными гранями. Пусть $\{P\}$ — множество точек, расположенных на поверхности S. В таком случае каждая тройка точек определяет плоскость и надлежащий выбор троек обеспечивает аппроксимацию поверхности S многогранником. К сожалению, во многих случаях число треугольников, необходимое для приемлемой аппрок-

симации, оказывается слишком большим и поэтому приходится обращаться к поверхностям высших порядков. Вследствие этого возникают проблемы представления таких кусочных поверхностей.

Одно из решений этой проблемы предусматривает задание на поверхности множества кривых и затем определение с помощью этих кривых *интерполяционных конечных* (участков) *поверхностей,* ограниченных соответствующими кривыми. Такие методы обсуждаются в разд. 13.4—13.6. Другой способ предусматривает использование *точек-ориентиров* или *плоскостей-ориентиров* таким образом, как это делалось на плоскости применительно к кривым Безье и *B*-сплайнам. Эти методы обсуждаются в разд. 13.7 и 13.8. В разд. 13.2 и 13.3 представлены некоторые из основных формул, используемых при изучении всех поверхностей такого рода. Воспроизведению поверхностей посвящен разд. 13.9, причем основное внимание уделено штриховке.

Важным аргументом при выборе способа математического описания для представления поверхности служит простота решения некоторых важных задач машинной графики, например определения пересечения двух поверхностей. Очевидно, что для этого больше всего подходят плоскости, однако при большом числе конечных участков поверхности, ограниченных замкнутыми кривыми, общий объем вычислений может оказаться значительным. Поверхности, построенные с помощью *B*-сплайнов, обладают свойством выпуклой оболочки (см. утверждение 11.2), что позволяет определить плоскости, ограничивающие расположение поверхности. Число таких плоскостей может оказаться значительно меньше, чем необходимо при аппроксимации многогранниками, и, следовательно, задачи на пересечение и подобные им легче решать, оперируя этими плоскостями. Дополнительные затраты, связанные с определением точного пересечения, часто очень малы. В самом деле, большинство выпуклых оболочек вообще не пересекаются, и, следовательно, отсутствует необходимость проверять, не пересекаются ли соответствующие поверхности. При составлении карт местности, ориентированных на машинную картографию, очевидно, целесообразно использовать плоскости, поскольку требования к гладкости изображения не очень высоки.

13.2. НЕКОТОРЫЕ ПРОСТЕЙШИЕ СВОЙСТВА ПОВЕРХНОСТЕЙ

В общем случае поверхность можно описывать с помощью одного из следующих уравнений:

Явное — $z = f(x,y)$; (13.1а)

Алгебраическое — $F(x,y,z) = Q$; (13.1*б*)

Параметрическое — $x=X(u,v)$, $y = Y(u,v)$, $z = Z(u,v)$. (13.1в)

Параметрическая форма задания поверхности удобнее всего в прикладных задачах машинной графики и в нашей книге она бу-

дет использоваться чаще всех остальных. При решении многих задач полезно располагать уравнением нормали к поверхности в точке. Последнее можно получить с помощью алгебраической формы задания поверхности. Поскольку $F(x, y, z)$ — константа, ее полный дифференциал равен нулю, так что

$$\frac{\partial F}{\partial x}\,dx + \frac{\partial F}{\partial y}\,dy + \frac{\partial F}{\partial z}\,dz = 0. \tag{13.2}$$

Величины dx, dy и dz обозначают вариации координат вдоль поверхности и поэтому можно считать, что они определяют вектор, являющийся касательной к рассматриваемой поверхности. В таком случае уравнение (13.2) можно интерпретировать как скалярное произведение вектора, компонентами которого служат частные производные, на вектор, являющийся касательной к рассматриваемой поверхности. Поскольку это произведение равно нулю, нормаль к поверхности параллельна первому вектору, (Вспомните обсуждение, проведенное нами в подразд. 10.7.1).

Пример 13.1. Задано уравнение плоскости $ax+by+cz=1$. Нормали к этой плоскости параллельны вектору (a, b, c).

Пример 13.2. Уравнение сферы с центром в начале координат имеет вид $x^2+y^2+z^2 = r^2$. Нормаль в каждой точке сферы (x, y, z) параллельна вектору (x, y, z).

При рассмотрении параметрического представления поверхностей будет использоваться следующее обозначение:

$$X_u = \partial X/\partial u. \tag{13.3}$$

Аналогичные обозначения вводятся и для других переменных. Поскольку векторы (X_u, Y_u, Z_u) и (X_v, Y_v, Z_v) являются касательными к поверхности, то, если (F_x, F_y, F_z) — нормальный вектор, справедливы

$$F_x X_u + F_y Y_u + F_z Z_u = 0, \tag{13.4а}$$

$$F_x X_v + F_y Y_v + F_z Z_v = 0. \tag{13.4б}$$

С помощью подстановки можно показать, что эти уравнения имеют решения:

$$F_x = Y_u Z_v - Z_u Y_v, \tag{13.5а}$$

$$F_y = -X_u Z_v + Z_u X_v, \tag{13.5б}$$

$$F_z = X_u Y_v - Y_u X_v. \tag{13.5в}$$

уравнения

Поскольку в системе уравнений (13.4) число неизвестных больше числа уравнений, ее решение неоднозначно и необходимо позаботиться о том, чтобы исключить все общие множители из правых частей уравнений (13.5). Эта мера необходима в связи с тем, что уравнения (13.4) определяют направление градиента, но не его значение. Этим обстоятельством можно воспользоваться при записи уравнения, связывающего градиенты, определяемые при параметрической и явной формах задания поверхности. Действительно, представление поверхности в явном виде является частным слу-

чаем алгебраического представления, в чем можно убедиться, переписав явное уравнение как

$$z - f(x, y) = 0. \tag{13.6}$$

Далее находим, что

$$F_x = -f_x, \quad F_y = -f_y, \quad F_z = 1. \tag{13.7}$$

Третье из уравнений (13.7) вводит ограничение, так что, в конечном счете, получаем

$$f_x = (Z_u Y_v - Z_v Y_u)/(X_u Y_v - X_v Y_u), \tag{13.8а}$$

$$f_y = (X_u Z_v - X_v Z_u)/(X_u Y_v - X_v Y_u). \tag{13.8б}$$

Пример 13.3. Параметрическое задание сферы с центром в начале координат имеет вид

$x = \cos u \cos v, \quad y = \sin u \cos v,$

$z = \sin v.$

Уравнение (13.5) после исключения общих множителей дает следующие значения для нормали:

$F_X = \cos u \cos v, \; F_v = \sin u \cos v,$

$F_z = \sin v,$

Соответствуют ли эти значения результатам примера 13.2? Получаем также, что $f_x = \cos u/\operatorname{tg} v$, $f_v = -\sin u/\operatorname{tg} v$.

13.3. ОСОБЫЕ ТОЧКИ ПОВЕРХНОСТИ

Если поверхность задана явным уравнением, то точки, в которых частные производные z по x и y обращаются в 0, называют *особыми,* или *стационарными.* Для изучения свойств этих точек воспользуемся разложением функции f в ряд Тейлора в их окрестности. Для частных производных будут применяться обозначения, аналогичные введенным в (13.3). Если x_0, y_0 — координаты особой точки, то можно записать:

$$f(x, y) = f(x_0, y_0) + (x - x_0) f_x + (y - y_0) f_y + \frac{1}{2} (x - x_0)^2 f_{xx} +$$

$$+ (x - x_0)(y - y_0) f_{xy} + \frac{1}{2} (y - y_0)^2 f_{yy} + \ldots, \tag{13.9}$$

где f_{xx} — вторая частная производная по x и т. п. В особой точке обе первые частные производные равны 0. Следовательно, вид функции определяется вторыми производными. Рассмотрим изменения этой функции относительно значения в точке (x_0, y_0) вдоль фиксированного направления. Обозначим через h отношение $(y - y_0)/(x - x_0)$. Тогда приведенное выше разложение в ряд Тейлора можно записать:

$$f(x, y) = f(x_0, y_0) + 2 (x - x_0)^2 [f_{xx} + 2 h f_{xy} + h^2 f_{yy}] + \ldots \tag{13.10}$$

(предполагается, что $x <> x_0$). Из элементарной алгебры известно, что знак квадратичного многочлена совпадает со знаком коэффициента при его старшем члене, если дискриминант его коэффици-

ентов отрицателен (многочлен не имеет действительных корней). Следовательно, знак выражения, заключенного в квадратные скобки (рассматриваемого как многочлен по h), будет всегда один и тот же, если

$$f_{xy}^2 - f_{xx} f_{yy} < 0. \qquad (13\ 11)$$

Очевидно, что из этого неравенства следует также, что такой знак имеют вторые частные производные f_{yy} и f_{xx}. Если они положительны, то выражение, заключенное в квадратные скобки, также положительно и, следовательно, значение функции вне особой точки больше ее значения в такой точке. Итак, особая точка представляет собой некоторый *минимум*. С другой стороны, если эти производные отрицательны, то особая точка представляет некоторый *максимум*. Если неравенство (13.11) не выполняется, то имеет место *седловая точка*. Все это означает, что существует направление, на котором особая точка образует минимум, и существует также другое направление, на котором эта особая точка образует максимум.

Пример 13.4. Рассмотрим функцию $z=f(x, y) =x^2+2axy+y^2$. Совершенно очевидно, что точка (0,0) — особая и выражение, заключенное в уравнении (13.10) в квадратные скобки, имеет в данном случае вид $2(1+2ah+h^2)$. Если абсолютное значение а меньше единицы, то особая точка представляет минимум Действительно, при малых значениях a рассматриваемая поверхность имеет вид чаши, поперечные сечения которой очень близки к окружностям Если a имеет большое положительное значение, то доминирующим становится член xy. Если знаки x и y совпадают, то функция будет принимать в основном положительные значения, а при разных знаках x и y будет наблюдаться противоположная картина Рассмотрим, в частности, прямую $x=y$. В таком случае $z = 2(1+a)x^2$ и при положительных значениях a пересечение поверхности и плоскости, проходящей через эту прямую, будет иметь в особой точке минимум. Если $x= - y$, то $z=2(1—a)x^2$ и, следовательно, аналогичное пересечение будет иметь в особой точке максимум. На рис. 13.1 изображена поверхность такого вида при $a = 2$. Эта иллюстрация объясняет, почему такие точки называют «седловыми».

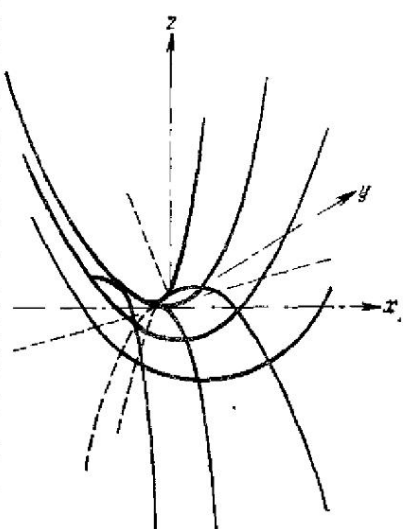


Рис. 13.1. Пример седловой точки

13.4. ЛИНЕЙНЫЕ И БИЛИНЕЙНЫЕ ИНТЕРПОЛЯЦИОННЫЕ УЧАСТКИ ПОВЕРХНОСТИ, ОГРАНИЧЕННЫЕ ЗАМКНУТЫМИ КРИВЫМИ

Простейшим средством интерполирования в трехмерном случае является плоский треугольник, заданный тремя точками $\{P_j\}_1^3$. Пусть u и v — скалярные переменные, принимающие зна-

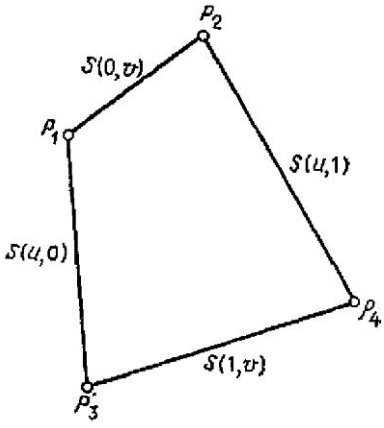


Рис. 13.2. Пример множества из четырех точек, определяющих некоторую билинейную поверхность

чения от 0 до 1. В таком случае поверхность треугольника, вершины которого находятся в указанных точках, определяется уравнением

$$T(u, v) = \mathbf{P}_1 u + \mathbf{P}_2 v + \mathbf{P}_3(1 - u - v) \qquad (13.12)$$

при условии, что $u + v <= 1$. Естественно, $T(1, 0) = P_1$, $T(0, 1) = P_2$ и $T(0, 0) = P_3$. Кроме того, $T(u, 0)$ — прямая, соединяющая точки P_1 и P_3, $T(0, v)$ — прямая, соединяющая точки P_2 и P_3, и $T(u, 1-u)$ — прямая, соединяющая точки P_1 и P_2.

Несколько более сложный вид имеет билинейная интерполяционная поверхность, задаваемая на множестве из четырех точек (рис. 13.2). Такая поверхность $S(u, v)$ определяется как

$$S(u, v) = \mathbf{P}_1 (1-u)(1-v) + \mathbf{P}_2 (1-u) v + \mathbf{P}_3 u (1-v) + \mathbf{P}_4 uv, \qquad (13.13)$$

где $0 <= u, v <= 1$.

Очевидно, что $S(0, 0) = P_1$, $S(0, 1) = P_2$ и т. д. Кроме того, $S(0, v)$ — отрезок прямой, соединяющий точки P_1 и P_2, $S(1, v)$ — отрезок прямой, соединяющий точки P_3 и P_4, и т. д. Если четыре заданные точки компланарны, то S представляет собой плоский интерполяционный четырехугольник; в противном случае S — поверхность второго порядка. Градиент интерполяционной поверхности по u и v можно найти с помощью обычного дифференцирования уравнения (13.13):

$$\partial S/\partial u = (\mathbf{P}_4 - \mathbf{P}_3) v + (\mathbf{P}_3 - \mathbf{P}_1)(1 - v), \qquad (13.14а)$$

$$\partial S/\partial v = (\mathbf{P}_4 - \mathbf{P}_3) u + (\mathbf{P}_2 - \mathbf{P}_1)(1 - u). \qquad (13.14б)$$

Отметим, что эти уравнения — векторные и, следовательно, если X_i обозначает координату x точки P_i, то

$$X^u = (X_4 - X_2) v + (X_3 - X_1)(1 - v),$$

аналогичные соотношения можно записать для Y^u и Z^u. Верхний индекс для обозначения частной производной здесь в отличие от выражения (13.3), где использован нижний индекс, введен для того, чтобы исключить путаницу с нижними индексами, обозначающими точки. Для определения градиента по координатам x, y можно воспользоваться уравнениями (13.8).

Пример 13.5. Рассмотрим точки $P_1 = (0,0,0)$, $P_2 = (0,1,0)$, $P_3 = (1,0,0)$ и $P_4 = (1,1,1)$. В таком случае координаты x, y и z каждой точки билинейной интерполяционной поверхности определяются следующими выражениями:

$$x(u, v) = u,\ y(u, v) = v,\ z(u, v) = uv \tag{13.15а}$$

или

$$z = xy\ ; \tag{13.15б}$$

градиент этой поверхности определяется как

$$\frac{\partial S}{\partial u} \begin{bmatrix} 1 \\ 0 \\ v \end{bmatrix}, \tag{13.16а}$$

$$\frac{\partial S}{\partial v} = \begin{bmatrix} 0 \\ 1 \\ u \end{bmatrix}. \tag{13.16б}$$

Объединяя уравнения **(13.15а), (13.16а) и (13.16*б*), получаем**

$$\partial z/\partial x = y, \tag{13.16в}$$

$$\partial z/\partial y = x, \tag{13.16г}$$

эти результаты можно было также получить при помощи непосредственного дифференцирования уравнения (13.15*б*).

Результаты последнего примера справедливы и для других случаев, поскольку всегда можно в качестве плоскости, заданной тремя точками, выбрать 2 = 0 и поместить одну точку в начало координат. Поверхность при этом будет определяться выражением $z = z_4 uv$.

13.5. ПЛАЗОВЫЕ ПОВЕРХНОСТИ

Плазовые поверхности получаются в результате применения процедуры линейного интерполирования к кривым, а не к точкам. Пусть $P(0, v)$ и $P(1, v)$ — уравнения, задающие соответственно две пространственные кривые. Тогда плазовая поверхность определяется как

$$S(u, v) = (1 - u)\, P(0, v) + u\, P(1, v). \tag{13.17}$$

Другими словами, такая поверхность порождается отрезком прямой, скользящим по двум кривым. Например, если этими двумя кривыми служат окружности, расположенные в плоскостях, перпендикулярных прямой, соединяющей их центры, то построенная таким образом поверхность представляет собой конус, усеченный двумя плоскостями[1].

[1] Название этого типа поверхностей связано с тем, что они используются при работе в плазе — площадке для разбиения в натуральных размерах чертежа судна или самолета, по которому изготовляют шаблоны для раскроя и выгиба элементов обшивки и набора корпуса. Иногда эти поверхности называют линейчатыми. — *Прим. перев.*

13.6. ПОВЕРХНОСТИ КУНСА

Эти поверхности обеспечивают интерполирование четырех кривых. Названы они по имени Кунса (S. A. Coons), который ввел их в обращение примерно в 1960-м году (см. разд. 13.10). Пусть $P(u, v)$ — функция двух переменных, обладающая таким свойством, что в случае, когда u или v — константа, эта функция сводится к параметрическому представлению пространственной кривой. Конечный участок поверхности с замкнутой границей можно построить, объединяя граничные кривые $P(u, 0)$, $P(u, 1)$, $P(0, v)$ и $P(1, v)$ следующим образом:

$$S(u, v) = P(u, 0)(1-v) + P(u, 1)v + P(0, v)(1-u) + P(1, v)u - \\ -P(0, 0)(1-u)(1-v) - P(0, 1)(1-u)v - P(1, 0)u(1-v) - \\ -P(1, 1)uv. \qquad (13.18)$$

Четыре последние члена необходимы для предотвращения того, чтобы попарные пересечения четырех кривых не учитывались дважды. Таким образом,

$$S(u, 0) = P(u, 0) + P(0, 0)(1-u) + P(1, 0)u - \\ -P(0, 0)(1-u) - P(1, 0)u = P(u, 0),$$

аналогичные выражения записываются для $S(u, 1)$ и т. д. Уравнение (13.18) гарантирует непрерывность построенной поверхности, которая, однако, не обязательно будет гладкой, что объясняется возможностью разрывности градиента. Прежде чем перейти к обсуждению способов обеспечения гладкости поверхности, введем более компактную форму записи уравнения (13.18). Пусть

$$\mathbf{P}_1(v) = \begin{bmatrix} P(0, v) \\ P(1, v) \end{bmatrix}, \qquad (13.19а)$$

$$\mathbf{P}_2(u) = \begin{bmatrix} P(u, 0) \\ P(u, 1) \end{bmatrix}, \qquad (13.19б)$$

$$\mathbf{b}(u) = \begin{bmatrix} 1-u \\ u \end{bmatrix}. \qquad (13.19в)$$

В результате уравнение (13.18) принимает вид

$$S(u, v) = \mathbf{b}'(u)\,\mathbf{P}_1(v) + \mathbf{P}_2'(u)\,\mathbf{b}(v) - \mathbf{b}'(u)\,M\,\mathbf{b}(v), \qquad (13.20)$$

где M — матрица с элементами $M_{ij} = P(i, j)$. До сих пор мы пользовались линейными функциями объединения, но можно ввести обобщение, позволяющее использовать в качестве объединяющих функции произвольного вида. Зададим, в частности, функцию

$$\mathbf{b}(u) = \begin{bmatrix} b_0(u) \\ b_1(u) \end{bmatrix} \qquad (13.21а)$$

$$b_i(u) = \delta_{iu} \tag{13.21б}$$

при любом целом значении u.

Пример 13.6. Рассмотрим объединяющие функции

$$b_0(u) = \cos^2\left(\frac{\pi}{2}u\right), \tag{13.22а}$$

$$b_1(u) = \sin^2\left(\frac{\pi}{2}u\right). \tag{13.22б}$$

Если границами служат прямые, пересекающиеся в точках, использовавшихся в примере 13.5, то координаты x и y определяются следующими уравнениями:

$$P(u, 0) = u P_3,\ P(u, 1) = u P_4 + (1-u) P_2,$$
$$P(0, v) = v P_2,\ P(1, v) = v P_4 + (1-v) P_3.$$

Уравнение поверхности Кунса, определяемой этими точками, имеет вид:

$$S(u, v) = \cos^2\left(\frac{\pi}{2}v\right) u P_3 + \sin^2\left(\frac{\pi}{2}v\right)[u P_4 + (1-u) P_2] +$$
$$+ \cos^2\left(\frac{\pi}{2}u\right) v P_2 + \sin^2\left(\frac{\pi}{2}u\right)[v P_4 + (1-v) P_3] -$$
$$-\cos^2\left(\frac{\pi}{2}u\right)\sin^2\left(\frac{\pi}{2}v\right) P_2 - \sin^2\left(\frac{\pi}{2}u\right)\cos^2\left(\frac{\pi}{2}v\right) P_3 -$$
$$-\sin^2\left(\frac{\pi}{2}u\right)\sin^2\left(\frac{\pi}{2}v\right) P_4.$$

В частности, координаты поверхности определяются уравнениями

$$x(u, v) = u, \quad y(u, v) = v, \tag{13.23а}$$

$$z(u, v) = \sin^2\left(\frac{\pi}{2}v\right) u + \sin^2\left(\frac{\pi}{2}u\right) v - \sin^2\left(\frac{\pi}{2}u\right)\sin^2\left(\frac{\pi}{2}v\right). \tag{13.23б}$$

для произвольных функций $b_i(u)$, обладающих тем свойством, что

Для обеспечения непрерывности градиента можно осуществлять интерполирование не только по кривым $P(u, v)$, но и по их производным. Для этого требуется в каждый вектор уравнений (13.19) ввести две дополнительные компоненты, представляющие производные и функции объединения для этих производных. Матрицу M также необходимо увеличить до размеров 4x4. Необходимость определять эти производные является недостатком данного метода, усложняющим процесс построения поверхности. Обычный формальный прием, применяемый в этом случае, предусматривает использование верхних индексов для обозначения производных; итак, для всякой функции $g(u, v)$ определяются

$$g^u = \partial g(u, v)/\partial u, \tag{13.24а}$$

$$g^{uv} = \partial^2 g(u, v)/(\partial u\, \partial v). \tag{13.24б}$$

Далее задается

$$\mathbf{P}_1(v) = \begin{bmatrix} P(0, v) \\ P(1, v) \\ P^u(0, v) \\ P^u(1, v) \end{bmatrix} \tag{13.25}$$

и аналогичное выражение для $P_2(u)$. Кроме того, задается

$$\mathbf{b}(u) = \begin{bmatrix} b_{00}(u) \\ b_{01}(u) \\ b_{10}(u) \\ b_{11}(u) \end{bmatrix} \tag{13.26}$$

с учетом того, что

$$b_{jt}^k(u) = \delta_{kj}\delta_{tu} \tag{13.27}$$

для целых значений u и $k = 0$, если соответствующая производная не берется, и $k = 1$, если дифференцирование производится. Таким образом,

$$b_{00}(0) = 1, \tag{13.28а}$$

$$b_{01}(1) = b_{00}^u(0) = b_{00}^u(1) = 0. \tag{13.28б}$$

Другими словами, $b_{0t}(u)$ — функция объединения для кривых и $b_{1t}(u)$ — функция объединения для угловых коэффициентов касательных. Значение δ_{kj} равно единице в двух случаях: $k = j = 0$, что соответствует первым двум компонентам вектора объединения, и $k=j=1$, что соответствует производным двух последних компонент. Матрица M теперь имеет вид

$$M = \begin{bmatrix} P(0, 0) & P(0, 1) & P^v(0, 0) & P^v(0, 1) \\ P(1, 0) & P(1, 1) & P^v(1, 0) & P^v(1, 1) \\ P^u(0, 0) & P^u(0, 1) & P^{uv}(0, 0) & P^{uv}(0, 1) \\ P^u(1, 0) & P^u(1, 1) & P^{uv}(1, 0) & P^{uv}(1, 1) \end{bmatrix}. \tag{13.29}$$

Уравнение (13.20) сохраняет справедливость в этих обозначениях. Более того, можно убедиться, что справедливы не только равенства вида $S(u, 0) = P(u, 0)$, но и вида $S^u(u, 0) = P^u(u, 0)$ и т. д.

13.7. ПОСТРОЕНИЕ ПОВЕРХНОСТЕЙ ПО ОРИЕНТИРАМ

Воспользовавшись обобщением многочленов Безье или B-сплайнов типов, рассмотренных в разд. 11.6, можно задать поверхность с помощью точек-ориентиров.

13.7.1. ПОВЕРХНОСТИ БЕЗЬЕ

Поверхность $S(u, v)$ можно определить как тензорное произведение $Q(u)R(v)$ двух многочленов Безье $Q(t)$ и $R(t)$, которое вычисляется отдельно по каждой координате точек-ориентиров. Эта процедура не влияет на вид переменных u и v, но отражается на коэффициентах. Задав $P^h_{ij} = Q^h{}_i R^h{}_j$, где h обозначает x, y или z, приходим к следующему уравнению поверхности:

$$S(u, v) = \sum_{i=0}^{m} \sum_{j=0}^{n} C_m^i u^i (1-u)^{m-i} C_n^j v^j (1-v)^{n-j} \mathbf{P}_{ij}. \qquad (13.30)$$

Легко показать, что

$$S(u, 0) = \sum_{i=0}^{m} C_m^i u^i (1-u)^{m-i} \mathbf{P}_{i0}; \qquad (13.31)$$

аналогичные выражения можно записать для $S(u, 1)$, $S(0, v)$, $5(1, v)$. Таким образом, поверхность $S(u, v)$ ограничена четырьмя многочленами Безье и угловыми являются точки P_{00}, P_{0n}, P_{m0} и P_{mn}. Воспользовавшись тем, что удвоенная сумма коэффициентов P_{ij} равна 1, можно продемонстрировать принадлежность построенной поверхности выпуклой оболочке точек-ориентиров.

Такую поверхность можно описать также, опираясь на ее участки, аналогично тому, как было показано в разд. 10.5 для многочленов Безье (см. теорему 10.1). Подобный способ описания удобно использовать при решении прикладных задач машинной графики в случаях, когда требуется разделить некоторый конечный участок поверхности, имеющий замкнутую границу, на меньшие участки для того, чтобы установить, какие элементы объекта являются видимыми, а какие — невидимыми (см. гл. 17).

13.7.2. ПОВЕРХНОСТИ, ПОСТРОЕННЫЕ С ПОМОЩЬЮ *B*-СПЛАЙНОВ

Поверхность, построенная с помощью B-сплайнов (B-сплайн-поверхность), формально определяется следующим образом:

$$\mathbf{P}(u, v) = \sum_{i=-m}^{k} \sum_{j=-m}^{l} N_{i,m}(u) N_{j,m}(v) \mathbf{P}_{ij} \qquad (13.32)$$

при $u_0 <= u <= u_{k+1}$ и $u_0 <= u <= u_{i+1}$. Без потери общности можно считать, что $u_0 = v_0 = 0$. Напомним, что $N_{i,m}(u)$ во всех случаях, за исключением $u_i < u < u_{i+m+1}$, и суммарное значение всех B-сплайнов в точке равно 1 (см. теорему 11.11). В таком случае получаем, что

$$\mathbf{P}(u, 0) = \sum_{i=-m}^{k} N_{i,m}(u) \sum_{j=-m}^{0} N_{j,m}(0) \mathbf{P}_{ij}.$$

Если все точки-ориентиры с отрицательными индексами точно такие же, как P_{i0}, то последнее уравнение упрощается:

$$\mathbf{P}(u,\ 0) = \sum_{i=-m}^{k} N_{i,\ m}(u)\,\mathbf{P}_{i\,0}, \qquad (13.33)$$

т. е. поверхность ограничена сплайном при $v = 0$. Использование этого метода с такими же допущениями приводит к получению идентичных результатов для остальных крайних значений u и v. Естественно, этот результат не имеет такого значения, как в случае поверхностей Кунса, поскольку уравнение (13.32) описывает не отдельный участок поверхности с замкнутой границей, а всю искомую поверхность. Конечные участки поверхности с замкнутой границей можно построить для всех множеств значений u и v, лежащих между точками склеивания: $u_i <= u <= u_{i+1}$, $v_i <= v <= v_{i+1}$. Полезно познакомиться с вариантами уравнения (13.33), соответствующими отдельным частным случаям.

Вначале рассмотрим билинейную интерполяционную поверхность при $m = 1$ и равномерном размещении точек склеивания (они отстоят друг от друга на единицу длины, т. е. в соответствии с соотношением (11.8) L=l). В таком случае,

$$\mathbf{P}(u,\ v) = (u - u_i)(v - v_j)\,\mathbf{P}_{ij} + (u - u_i)(v_{j+1} - v)\,\mathbf{P}_{i,\ j-1} +$$
$$+ (u_{i+1} - u)(v - v_j)\,\mathbf{P}_{i-1,\ j} + (u_{i+1} - u)(v_{j+1} - v)\,\mathbf{P}_{i-1,\ j-1}. \qquad (13.34)$$

Итак, получена билинейная форма, аналогичная той, что определяется уравнением (13.13). Поскольку $u_{i+1} - u_i = 1$ и $v_{j+1} - v_j = 1$, мы получаем,

$$\mathbf{P}(u_i,\ v) = (v - v_j)\,\mathbf{P}_{i-1,\ j} + (v_{j+1} - v)\,\mathbf{P}_{i-1,\ j-1}, \qquad (13.34а)$$
$$\mathbf{P}(u_{i+1},\ v) = (v - v_j)\,\mathbf{P}_{ij} + (v_{j+1} - v)\,\mathbf{P}_{i,\ j-1} \qquad (13.34б)$$

и т. д. Кроме того,

$$\mathbf{P}(u_i\ v_j) = \mathbf{P}_{i-1,\ j-1}, \qquad (13.35а)$$
$$\mathbf{P}(u_{i+1},\ v_{j+1}) = \mathbf{P}_{ij} \qquad (13.35б)$$

что

и т. д. Следовательно, билинейная интерполяционная поверхность является частным случаем поверхности (m=1), определяемой уравнением (13.32).

Перейдем теперь к поверхностям, получаемым с помощью равномерных B-сплайнов второго порядка $(m = 2)$ при L = 1. Зададим снова $u_0 = 0$, так что $u_i = i$. В разд. 11.7 (см. пример 11.4) было установлено, что такие сплайны особенно хорошо подходят для описания формы объекта. Анализ, позволивший вывести уравнения (11.30а) и (11.32), можно повторить, в данном случае с тем, чтобы получить подобные выражения. В частности, мы находим, что

$$\mathbf{P}(u_i,\ v_j) = \frac{1}{4}(\mathbf{P}_{i-1,\ j-1} + \mathbf{P}_{i-1,\ j-2} + \mathbf{P}_{i-2,\ j-1} + \mathbf{P}_{i-2,\ j-2}), \qquad (13.36)$$

т. е. поверхность проходит через центр четырехугольника, образованного четырьмя точками-ориентирами. Кроме того,

$$\frac{\partial}{\partial u}\mathbf{P}|_{u_i v_i} = \frac{1}{2}(\mathbf{P}_{i-1,\ j-1} + \mathbf{P}_{i-1,\ j-2}) -$$

$$-\frac{1}{2}(\mathbf{P}_{i-2,\ j-1} + \mathbf{P}_{i-2,\ j-2}). \qquad (13.37)$$

Из последних двух уравнений следует, что, если имеется многогранник, гранями которого являются четырехугольники, а вершины могут быть отображены на некоторую квадратную сетку, то сплайн-поверхность, при построении которой вершины многогранника использовались в качестве точек-ориентиров, обладает следующим свойством: она проходит через центры всех граней многогранника и является касательной к нему в этих точках.

13.8. ВЫБОР РАЗБИЕНИЯ ПОВЕРХНОСТИ

Если поверхность задается множеством точек, необходимо выбрать соответствующее разбиение или определить множество точек-ориентиров прежде, чем приступить к применению любого из описанных в четырех предыдущих разделах методов. Эта задача является обобщением задачи о размещении узлов, рассмотренной в разд. 12.4. Учитывая трудность последней, не удивительно, что автоматические методы решения задачи, сформулированной в данном разделе, не известны. При решении многих прикладных задач машинной графики кривые вычерчиваются на соответствующем макете, а именно они и служат исходными данными для дискретизации. Отыскание разбиения при обработке изображений эквивалентно решению задачи сегментации (см. гл. 4). Поскольку в этом случае условия непрерывности не имеют значения, задача несколько упрощается.

13.9. ВОСПРОИЗВЕДЕНИЕ ПОВЕРХНОСТЕЙ И ШТРИХОВКА

Основная проблема, возникающая при воспроизведении изображений кривых, заключается в преобразовании математического описания соответствующей кривой в набор координат пикселов. Это — единственная информация, которая необходима для воспроизведения плоской кривой. В случае же поверхности одних лишь координат (x_i, y_i, z_i) ее точек явно недостаточно для того, чтобы воспроизвести ее изображение. Известны два основных подхода: кодирование информации о высоте (координаты z) с помощью цвета и штриховка. Применение цветового кодирования приводит к воспроизведению контуров, что может быть полезно при решении некоторых прикладных задач, но не привлекательно ни эстетически, ни информативно. Использование штриховки позволяет получать изображения лучшего качества, однако требует

большего объема вычислений. Если все поверхности являются плоскостями, то эффект штриховки можно получить, окрашивая в один цвет или воспроизводя в одном уровне серого тона проекцию каждой видимой плоскости. Иногда этого оказывается достаточно для создания впечатления глубины изображения. Ряд проблем, осложняющих воспроизведение неплоских поверхностей, возникает из-за необходимости работать с их проекциями на некоторую плоскость, и, следовательно, решать задачу разделения видимых и невидимых элементов — дело в том, что одни точки поверхности могут заслонять другие. В частности, очертания проекции конечного участка поверхности с замкнутой границей не всегда совпадают с проекцией этой границы. Поскольку исчерпывающее обсуждение этих проблем выходит за пределы задач нашей книги, мы ограничимся здесь изложением лишь основных принципов.

При использовании растровых графических устройств для воспроизведения контуров достаточно просто установить некоторое соответствие между координатой z и цветом или уровнем серого тона. Если плотность выборки точек достаточно высока, то контуры, соответствующие постоянному значению z, будут представляться кривыми постоянной яркости. При использовании векторных графических устройств эта задача становится существенно более трудной, так как соответствующие кривые приходится прослеживать на воспроизводимой поверхности.

Использование штриховки подразумевает наличие некоторого источника света, а также сведений об отражательных способностях воспроизводимой поверхности. Если источник света удален от освещаемого объекта на очень значительное расстояние, то можно считать, что все световые лучи параллельны и вид поверхности определяется исключительно ее собственными свойствами и ориентацией относительно источника света. Количество света, падающего на поверхность, пропорционально скалярному произведению нормали n к поверхности и вектора 1, параллельного направлению освещения (рис. 13.3). (В разд. 13.2 было показано, как строятся нормали к поверхностям, задаваемым математическими описаниями.) Если поверхность является абсолютным зеркалом, то для вычисления яркости каждой ее точки можно воспользоваться фундаментальным законом отражения света: угол отражения равен углу падения. Наблюдатели могут видеть свет лишь в том случае, если их взгляд направлен должным образом. Пусть e— вектор,

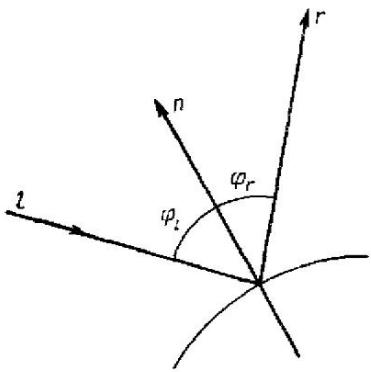


Рис. 13 3. Отражение света от поверхности:
n — нормаль к поверхности, 1 — направление освещения источником света, r—направление отраженного света φ_i — угол падения, φ_r — угол отражения света

параллельный этому правильному направлению; допустим также, что этот вектор, как и вектор 1, нормирован таким образом, что длина каждого равна единице. В таком случае биссектриса образуемого ими угла параллельна их сумме 1 + e и, следовательно, количество света, доходящего до наблюдателя, пропорционально скалярному произведению этого вектора и нормали n к поверхности. Поверхность имеет наиболее яркий вид при равенстве указанного угла нулю и кажется черной при равенстве угла 90°. Этот тип отражения называется *зеркальным* и он лишь частично определяет общую яркость поверхности. Остальная яркость определяется *диффузным* отражением света. Это подразумевает, что отражение света одинаково по всем направлениям и различия в яркости определяются исключительно различиями количества падающего света. Следовательно, яркость, определяемая диффузным отражением, пропорциональна скалярному произведению векторов l и n. Очевидно, что при отрицательности скалярного произведения отраженный свет отсутствует полностью, поэтому нас должны интересовать лишь положительные значения этого скалярного произведения. Обозначив через I_s зеркальное отражение и через Id диффузное отражение, запишем следующие соотношения:

$$I_s = \max [0,\ \mathbf{e}\cdot\mathbf{n}], \tag{13.38а}$$

$$I_d = \max [0,\ \mathbf{l}\cdot\mathbf{n}]. \tag{13.38б}$$

Для определения значения общей яркости необходимо использовать сумму этих величин, взятых с некоторыми весами. Кроме того, имеются эмпирические данные, указывающие, что возведение I_s в степень k позволяет придать большую реалистичность передаче наиболее ярких частей изображения, что оказывается недостижимым в противном случае (см. разд. 13.10). Итак,

$$I = m I_d + g I_s^k, \tag{13 39}$$

где m — коэффициент, учитывающий матовость поверхности, и g — коэффициент, учитывающий глянцевитость поверхности. Естественно, важно соотношение значений коэффициентов m и g. Некоторые авторы предлагают выбирать их, исходя из соотношения $m + g = 1$. Имеется публикация, свидетельствующая о том, что выбор показателя степени k в диапазоне 5 — 60 позволил получить очень выразительные изображения [13.2].

При практическом использовании приведенных формул необходимо определять нормаль в каждой точке воспроизводимой поверхности. Имеются сведения о различных способах приближения, предусматривающих вычисление значений яркости лишь в небольшом числе точек и последующее определение яркостей остальных точек при помощи интерполирования (см. разд. 13.10).

Пример 13.7. Воспользуемся выражением, определяющим нормаль к сфере, которое было получено в примерах 13.2 и 13.3, для построения заштрихованного изображения сферы. Будем считать, что соответствующая сфера освещается сверху, т, е. $l = (0,0,1)$, что наблюдатель смотрит на сферу по направ-

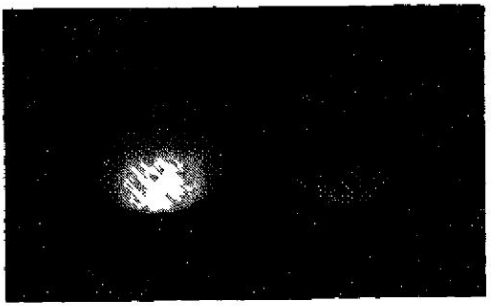


Рис. 13.4. Заштрихованные изображения сферы, полученные с помощью метода, который изложен в примере 13.7:
слева — матовая сфера (g/m=l/10), *справа* — глянцевитая (g/m=10/l)

лению оси x и что изображение воспроизводится в плоскости y-z. В таком случае $\mathbf{e}=(1,0,0)$ и нормированная сумма этих двух векторов равна $(1/\sqrt{2}, 0, 1/\sqrt{2})$. Так как $\mathbf{n}=(x, y, z)$, находим, что

$$I_s = \max [0, (x+z)/\sqrt{2}], \quad I_d = \max [0, z].$$

Из уравнения, определяющего сферу, находим, что $x=\pm\sqrt{1-y^2+z^2}$, и, следовательно, уравнение (13 39) принимает вид

$$I = m \max [0, z] + g [\max [0, z \pm \sqrt{1-y^2+z^2}]]^k. \qquad (13.40)$$

Это выражение нетрудно вычислить для различных значений y и z, удовлетворяющих условию $y^2+z^2 \leqslant 1$ (см задачу 13 3). На рис. 13.4 приведены два изображения сферы, в обоих случаях использовано $k=1$.

13.10. БИБЛИОГРАФИЧЕСКИЕ ЗАМЕЧАНИЯ

Проблемы воспроизведения и описания поверхностей освещаются в литературе, главным образом начиная с 1970 года. Рост популярности этой проблемы совпадает с появлением недорогих растровых графических устройств. Более ранние работы были в основном посвящены математическому описанию поверхностей[1]. Один из первых обзоров, посвященных поверхностям Кунса, дан в статье [13.4]. Результаты исследования, посвященного поверхностям Безье и B-сплайн-поверхностям, отражены в статье [13.8]. В сборнике [11.2], а также в статьях [13.7, 13.10] обсуждаются различные аспекты использования поверхностей при решении прикладных задач машинной графики. B-сплайн-поверхности, в связи с чрезвычайно локальным характером их представления, представляются наилучшим средством для тех прикладных задач, при решении которых поверхность может строиться в интерактивном режиме. Диссертация [13.1] является прекрасным источником сведений, касающихся воспроизведения изображений поверхностей, и (разд. 13.7 написан по материалам этой работы. В статье [13.2] помещена небольшая часть материала из [13.1], однако в ней приведено множество примеров и к тому же этот источник легко доступен. Модель, представленная уравнениями (13.38),

[1] Художники, конечно, уже много столетий занимаются проблемами описания и штриховки поверхностей, однако их методы непригодны для непосредственного использования при работе с ЭВМ.

(13.39), предложена в статье [13.9]. В работе [13.11] приведена более общая модель, учитывающая прошедший свет, а в статье [13.3] — модель, учитывающая зависимость отражательной способности от длины световой волны. В статье [13.7] приведено всестороннее обсуждение задачи воспроизведения изображения поверхности, включая методы определения нормали, разделения видимых и невидимых элементов и штриховки. Статья [13.6] посвящена родственной теме — штриховке при составлении географических карт. Следует подчеркнуть, что правила штриховки, используемые художниками, отличаются от формальных правил, описанных в разд. 13.9. Они часто предполагают, что рассеянное освещение определяется вторичными отражениями источника света и что самой темной областью является участок, лежащий между самой светлой частью изображения, определяемой зеркальным отражением, и максимумом диффузного отражения (см., например, монографию [13.5]).

13.11. ЗАДАЧИ

13.1. Покажите, что уравнения, определяющие нормаль к сфере, которые получены в примерах 13.2 и 13.3, эквивалентны.

13.2. Проанализируйте форму конечного участка поверхности с замкнутой границей из примера 13.5, воспользовавшись поперечными сечениями поверхности $S(u, v)$, такими, что $S(u, u)$, $S(u, 1 - u)$, $S(u, 1/3)$, $S(u, 1/2)$, $S(u, 2/3)$ и т. д.

13.3. а. Напишите программу, реализующую уравнение (13.40), и используйте ее для получения изображения заштрихованной сферы типа приведенного на рис. 13.4. Изучите влияние изменения параметров m, g и k. б. Покажите, что годографы точек с одинаковой яркостью, определяемой зеркальным отражением, представляют собой эллипсы. Дайте строгое доказательство того факта, что все такие эллипсы расположены внутри окружности, являющейся проекцией сферы.

Глава 14

МАТЕМАТИЧЕСКИЕ АСПЕКТЫ ДВУХМЕРНОЙ МАШИННОЙ ГРАФИКИ

14.1. ВВЕДЕНИЕ

Основной задачей машинной графики является воспроизведение изображений на экране, исходя из их математических описаний. Для этого требуется создание (обычно с помощью главной ЭВМ) файла воспроизведения визуального отображения, который затем передается в устройство отображения графической информации. В векторной графике для управления устройствами отображения требуются команды типа «провести вектор», и поэтому содержимое файла воспроизведения визуальных отображений сос

тавляют главным образом элементарные команды. В растровой графике команды типа «провести вектор» не относятся к категории элементарных и устройство отображения преобразовывает их в соответствующие конфигурации пикселов, хранящиеся в памяти обновления изображения. Строго говоря, большинство растровых графических устройств не располагают файлом воспроизведения визуального отображения. Целесообразно, однако, описывать работу этих устройств, оперируя этим файлом, независимо от того, каким именно образом устройство отображения его формирует. При воспроизведении очень простых изображений формирование файла воспроизведения визуального отображения тривиально, однако по мере роста сложности искомых отображений эта задача становится все более и более трудной.

Пример 14.1. Для того чтобы воспроизвести на экране дисплея изображение квадрата с яркостью L и длиной стороны s, разместив одну из его вершин в точке (x_0, y_0) и ориентировав стороны параллельно вертикальной и горизонтальной осям координат, можно воспользоваться следующей последовательностью команд из набора, приведенного в разд. 1.7 (см. табл. 1.1):

fore (L) *setp* (x_0, y_0)

vec$(x_0 + s, y_0)$ *vec* $(x_0 +$

$s, y_0 + s)$ *vec* $(x_0, y_0 + s)$

vec (x_0, y_0)

Эту группу команд можно рассматривать как некоторую процедуру *square* (x_0, y_0, s, L) (квадрат). Для устранения изображения такого квадрата можно обратиться к процедуре *square* $(x_0, y_0, s, 0)$. (Ниже рассматриваются возможные побочные эффекты стирания.) Более трудной задачей является заполнение квадрата произвольной ориентации определенным цветом, для чего необходимо воспользоваться одним из алгоритмов, рассмотренных в гл. 8. В то же время квадрат, стороны которого расположены вертикально и горизонтально, можно интерпретировать как фрагмент изображения (окно) и для воспроизведения и заполнения изображения можно воспользоваться следующей программой:

back (L)

win $(x_0, y_0, x_0 + s, y_0 + s)$

erase

Мы будем называть ее программой *fsquare* (x_0, y_0, s, L).

Этот пример иллюстрирует еще одну проблему, которая обычно возникает при использовании устройств отображения графической информации. Данный набор команд не определяет, является ли заданная вершина (x_0, y_0) верхней левой, нижней левой, верхней правой или нижней правой. Ответ определяется типом графического дисплея! В некоторых из них предполагается, что начало координат (0, 0) помещается в нижнем левом углу, а в других — в верхнем правом. В некоторых случаях начало координат можно задавать с помощью какой-либо графической команды, в других же пользователь должен формировать файлы, учитывая начало координат, определяемое собственно устройством.

Пример 14.2. С помощью следующей простой программы можно воспроизвести изображение, создающее иллюзию квадрата, удаляющегося от наблюдателя.

Алгоритм 14.1. Движущийся квадрат

1. While s — положительно do:
   Begin.
2. *square (x, y, s, L)*
3. sleep (T)
4. *square (x, y, s,* 0)
5. Значение s уменьшается на 2 и L — на 1, значения x *к* y увеличиваются на 1.
   End.
6. Конец алгоритма.

Шаг 2 обеспечивает воспроизведение изображения квадрата со стороной s и цветом L, а шаг 4 обеспечивает воспроизведение изображения квадрата аналогичного размера и цвета 0, т. е. стирание изображения квадрата, воспроизведенного на шаге 2[1]. Команда *sleep (T)* (замри) дает программе указание прекратить исполнение на T единиц времени для того, чтобы квадрат можно было увидеть прежде, чем он будет стерт. Во многих системах с разделением времени суммарное время ожидания и выполнения команд достаточно велико и поэтому дополнительные задержки не нужны. Вместо процедуры *square* можно использовать процедуру *fsquare* для того, чтобы получить такой же результат для заполненного квадрата. При программировании более сложных перемещений необходимо изменить шаг 5. Если, однако, требуется предусмотреть повороты, то необходимо изменить основную программу воспроизведения изображения квадрата, поскольку при поворотах стороны квадраты перестают быть параллельными осям координат. Еще одна проблема может возникнуть в связи с тем, что квадрат при перемещении может выйти за пределы экрана. За этим необходимо следить, и соответствующие команды следует изменить таким образом, чтобы воспроизводилась лишь та часть квадрата, которая находится в пределах допустимых границ воспроизводимого изображения. Таким образом, часть квадрата должна быть отсечена. (Выполнение отсечения нельзя возлагать на устройство воспроизведения отображения. Более подробные сведения по этому поводу приведены в разд. 15.1.)

Пример 14.3. Требуется воспроизвести изображение куба таким образом, чтобы одна из его вершин имела координаты x_0, y_0, z_0, а длина ребра была равна s. Необходимо построить проекцию трехмерного объекта на плоскость. Если строится движущееся изображение, то может потребоваться много таких проекций. Если требуется построить штриховое изображение, то для его воспроизведения можно воспользоваться простой программой, аналогичной процедуре *square*. Для воспроизведения заполненного изображения необходимо прежде установить, какие три из шести граней куба являются видимыми. Если воспроизводится изображение более чем одного куба, то следует выяснить, не заслоняет ли один из них другой, и, следовательно, возникает даже более сложная задача разделения видимых и невидимых элементов объекта.

Приведенные примеры иллюстрируют некоторые основные задачи, которые необходимо решать при воспроизведении изображе-

[1] Реализация команды «стирание» не является тривиальной операцией. (Вспомните наше обсуждение этой проблемы в разд. 1.5, 1.6.) В векторных графических устройствах более эффективной является непосредственная обработка списка воспроизводимых объектов. Естественно, это делает невозможным перенос процедур с векторных на растровые графические устройства. При работе с последними можно либо использовать команду *enable,* чтобы предотвратить стирание фона и прочих объектов, либо задавать некоторую окрестность объекта (окно), осуществлять в ней полное стирание и затем восстанавливать сцену. Естественно, формирование и преобразование сложных изображений требуют тщательного ведения учета.

ния. Для поворота изображения и получения проекций необходимо осуществлять *преобразование координат,* что, в свою очередь, требует проверять, находятся ли объекты по-прежнему в поле зрения и не заслоняют ли они друг друга. В целом эти задачи можно определить как *проблему видности* (наблюдаемости) объектов на изображении. В этой и следующей главах главным образом будет обсуждаться воспроизведение изображений двухмерных объектов, трехмерная машинная графика является предметом рассмотрения в гл. 16 и 17. Преобразованию двухмерных координат посвящен разд. 14.2. В разд. 14.3 и 14.4 приведен ряд математических результатов, используемых при решении значительного числа задач машинной графики. Процесс установления пересечения прямой (или многоугольника) с другим многоугольником обычно называют *отсечением* (сечением) — он будет предметом нашего рассмотрения в гл. 15.

Алгоритмы, описываемые в данной главе, с математической точки зрения очень просты и основываются исключительно на элементарных сведениях из аналитической геометрии. Эти алгоритмы, однако, образуют основу для более сложных алгоритмов, обеспечивающих воспроизведение интересных изображений.

14.2. ДВУХМЕРНЫЕ ПРЕОБРАЗОВАНИЯ

Очень распространенная задача машинной графики заключается в определении местоположения пиксела (x, y) после его поворота относительно точки (x_0, y_0) на угол θ. Пусть r — длина вектора, соединяющего точку (x, y) с точкой (x_0, y_0). Поскольку ее значение при повороте не изменяется, справедливы следующие уравнения (обозначения расшифровываются на рис. 14.1):

$$x - x_0 = r\cos\varphi, \quad y - y_0 = r\sin\varphi; \tag{14.1а}$$

$$X - x_0 = r\cos(\varphi + \theta), \quad Y - y_0 = r\sin(\varphi + \theta). \tag{14.1б}$$

Разложив косинус и синус суммы двух углов в уравнении (14.1$б$) и осуществив подстановку выражений для тригонометрических функций из уравнения (14.1 а), получаем уравнения, определяющие значения

$$X = x_0 + (x - x_0)\cos\theta - (y - y_0)\sin\theta, \tag{14.2а}$$

$$Y = y_0 + (x - x_0)\sin\theta + (y - y_0)\cos\theta. \tag{14.2б}$$

Эти уравнения можно представить в более простом виде, определив *матрицу поворота* относительно начала координат

$$R(\theta) = \begin{bmatrix} \cos\theta & -\sin\theta \\ \sin\theta & \cos\theta \end{bmatrix} \tag{14.3а}$$

и векторы

$$\mathbf{X} = \begin{bmatrix} X \\ Y \end{bmatrix} \quad \mathbf{x} = \begin{bmatrix} x \\ y \end{bmatrix} \quad \mathbf{x}_0 = \begin{bmatrix} x_0 \\ y_0 \end{bmatrix}. \tag{14.3б}$$

(X, Y) новых координат через старые:

В таком случае

$$\mathbf{X} = \mathbf{x}_0 + R(\theta)(\mathbf{x} - \mathbf{x}_0). \qquad (14.4)$$

Перенос точки можно легко учесть, прибавив к значениям соответствующих координат величину перемещения. *Изменение масштаба* производится при помощи умножения значений координат на масштабный множитель. Эту операцию можно определить

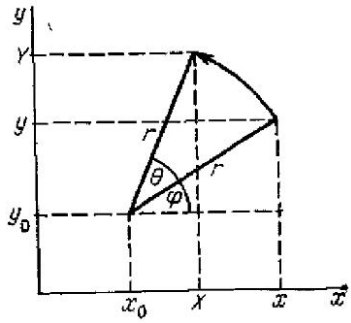


Рис. 14 1. Расшифровка обозначений, входящих в выражение, связывающее новые координаты *X, Y* со старыми *x, y* при повороте пиксела на угол 9 относительно точки (x_0, y_0)

в матричном виде, задав соответствующую матрицу масштабных множителей:

$$S(S_x, S_y) = \begin{bmatrix} S_x & 0 \\ 0 & S_y \end{bmatrix}. \qquad (14.5)$$

Для единообразия зададим и переносы матрицами, введя в векторы, определяемые уравнением (14.3*б*), третью компоненту, равную 1. Матрицы поворота и изменения масштаба можно модифицировать, введя в каждую из них третью строку и третий столбец; первые два элемента этой строки равны 0, а третий — 1. В результате матрица поворота принимает вид

$$T(\Delta x, \Delta y) = \begin{bmatrix} 1 & 0 & \Delta x \\ 0 & 1 & \Delta y \\ 0 & 0 & 1 \end{bmatrix} \qquad (14.6)$$

Теперь объединим эти матрицы в единую матрицу преобразований. Действительно, выполнение последовательности преобразований соответствует перемножению соответствующих матриц. Можно убедиться в том, что произведение матрицы $T(\bullet x, \bullet y)$ на расширенную матрицу $R(\theta)$ равно

$$\begin{bmatrix} \cos\theta & -\sin\theta & \Delta x \\ \sin\theta & \cos\theta & \Delta y \\ 0 & 0 & 1 \end{bmatrix}. \qquad (14.7а)$$

Произведение расширенной матрицы поворота $R(\theta)$ на матрицу $T(\bullet x, \bullet y)$ равно

$$\begin{bmatrix} \cos\theta & -\sin\theta & \Delta x\cos\theta - \Delta y\sin\theta \\ \sin\theta & \cos\theta & \Delta x\sin\theta + \Delta y\cos\theta \\ 0 & 0 & 1 \end{bmatrix}. \qquad (14.7б)$$

Уравнение (14.7а) определяет матрицу преобразования, состоящего в переносе, за которым следует поворот. Уравнение (14.7*б*) определяет матрицу преобразования, состоящего в повороте, за которым следует перенос. Поворот относительно точки (x_0, y_0) можно представить так же, как последовательность переносов и поворотов относительно начала координат:

$$\mathbf{X} = T(x_0,\ y_0)\,R(\theta)\,T(-x_0,\ -y_0)\,\mathbf{x}. \qquad (14.8)$$

В результате матрица поворота относительно точки (x_0, y_0) на угол θ, за которым следует перенос $(\bullet x, \bullet y)$, определяется как

$$M(x_0,\ y_0,\ \theta,\ \Delta x,\ \Delta y) = \begin{bmatrix} \cos\theta & -\sin\theta & \Delta x + x_0(1-\cos\theta) - y_0\sin\theta \\ \sin\theta & \cos\theta & \Delta y + y_0(1-\sin\theta) - x_0\cos\theta \\ 0 & 0 & 1 \end{bmatrix}. \qquad (14.9а)$$

Таким образом,

$$\mathbf{X} = M(x_0,\ y_0,\ \theta,\ \Delta x,\ \Delta y)\,\mathbf{x}. \qquad (14.9б)$$

Уравнения (14.9) определяют общий вид преобразования, обеспечивающего изменение местоположения точки на плоскости. Следует подчеркнуть, что обратную матрицу для матрицы $M(x_0, y_0, \theta, \bullet x, \bullet y)$ можно легко получить, заменив $\bullet x$ и $\bullet y$ на $-\bullet x$ и $-\bullet y$, θ на $-\theta$, изменив порядок операций в уравнении (14.8) на обратный.

14.3. ОДНОРОДНЫЕ КООРДИНАТЫ

Представление положения точки на плоскости с помощью трехкомпонентного вектора можно использовать для внутреннего изменения масштаба в процессе определения координат точки. Допустим, что мы изменяем единицу измерения координат x и y. В таком случае значение третьей компоненты вектора будет равно не 1, а масштабному множителю h и, следовательно, мы можем записать, что

$$\mathbf{x} = \begin{bmatrix} hx \\ hy \\ h \end{bmatrix}. \qquad (14.10)$$

Координаты (hx, hy, h) называются *однородными координатами* точки. Очевидно, что ни одна из матриц преобразований не должна подвергаться изменениям, поскольку новое представление соответствует умножению на масштабный множитель[1]. Основное преимущество такого представления становится очевидным при

[1] Отметим, что такое умножение не приводит к изменению масштаба воспроизводимого изображения. Масштабный множитель Л будет исключен до формирования воспроизводимого изображения.

обращении к уравнениям прямых. Стандартное уравнение, определяющее прямую с угловым коэффициентом μ, имеет вид

$$y = \mu x + \nu. \tag{14.11а}$$

Уравнение в такой форме нельзя использовать для описания прямых, параллельных оси y. С другой стороны, уравнение вида

$$a\xi + b\eta + c\zeta = 0. \tag{14.12а}$$

Определив a как вектор-столбец с элементами a, b и c, уравнение прямой можно представить в виде скалярного произведения

$$\mathbf{a}' \mathbf{x} = 0. \tag{14.12б}$$

допускает описание таких прямых посредством задания $b = 0$. Умножив обе части уравнения (14.11*б*) на масштабный множитель h и задав

$$\xi = xh, \quad \eta = yh, \quad \zeta = h,$$

приходим к уравнению прямой в однородных координатах.

Если к множеству точек применено преобразование (M — матрица, задающая это преобразование), то точки, бывшие коллинеарными до преобразования, должны сохранить коллинеарность. Если A — вектор-столбец, в который отображается вектор-столбец a, то должно быть справедливо уравнение

$$\mathbf{A}' \mathbf{X} = 0, \tag{14.12в}$$

и, следовательно,

$$\mathbf{A} = (M')^{-1} \mathbf{a}. \tag{14.13}$$

Отметим также, что некоторую бесконечно удаленную точку можно представить, задав для ξ, нулевое значение. Возможность работать с бесконечно удаленными точками точно таким же образом, как со всеми остальными точками, дает то преимущество, что упрощается обработка точек, координаты которых выходят за пределы доступного диапазона числовой точности (переполнение или исчезновение значащих разрядов). Так, например, прямая может задаваться двумя точками, координаты которых лежат за пределами указанного диапазона, и в то же время проходит через точки, попадающие в этот диапазон и подлежащие воспроизведению. При использовании однородных координат можно с помощью изменения масштаба ввести значения координат точек, задающих прямую, в приемлемые пределы. После этого для определения видимой части прямой можно использовать процедуру отсечения (см. гл. 15).

Проиллюстрируем ряд математических достоинств однородных координат при выводе нескольких полезных формул.

14.3.1. УРАВНЕНИЕ ПРЯМОЙ, ЗАДАВАЕМОЙ ДВУМЯ ТОЧКАМИ

Пусть координаты точек P_1 и P_2 равны (x_1, y_1, w_1) и (x_2, y_2, w_2) соответственно. Точка с координатами (x, y, w) будет коллинеарна точкам P_1 и P_2, если ее координаты линейно зависимы от координат этих точек. Это означает, что определитель матрицы, столбцы которой задают координаты трех рассматриваемых точек, должен быть равен 0, т. е.

$$\det \begin{bmatrix} x & x_1 & x_2 \\ y & y_1 & y_2 \\ w & w_1 & w_2 \end{bmatrix} = 0. \tag{14.14}$$

Это уравнение задает прямую, проходящую через точки P_1 и P_2. Раскрыв определитель, мы можем записать это уравнение в более привычном виде:

$$x(y_1 w_2 - w_1 y_2) + y(w_1 x_2 - x_1 w_2) + w(x_1 y_2 - y_1 x_2) = 0. \tag{14.14'}$$

[Отметим, что последнее уравнение эквивалентно уравнению (10.2).]

14.3.2. КООРДИНАТЫ ТОЧКИ, ОПРЕДЕЛЯЕМОЙ ПЕРЕСЕЧЕНИЕМ ДВУХ ПРЯМЫХ

Аналогичным образом можно определить точку, лежащую на пересечении прямых L_1 и L_2 с коэффициентами (a_1, b_1, c_1) и (a_2, b_2, c_2) соответственно. Поскольку всякая третья прямая с коэффициентами (a, b, c), проходящая через эту точку, должна быть линейно зависимой от первых двух прямых, определитель, столбцы которого составлены из коэффициентов этих прямых, должен быть равен 0. Итак,

$$\det \begin{bmatrix} a & a_1 & a_2 \\ b & b_1 & b_2 \\ c & c_1 & c_2 \end{bmatrix} = 0. \tag{14.15}$$

В результате разложения определителя получаем

$$a(b_1 c_2 - c_1 b_2) + b(c_1 a_2 - a_1 c_2) + c(a_1 b_2 - b_1 a_2) = 0. \tag{14.15'}$$

Поскольку (a, b, c) представляют собой коэффициенты произвольной прямой, проходящей через точку пересечения прямых L_1 и L_2, члены, стоящие в последнем уравнении в скобках, указывают координаты этой точки пересечения.

14.3.3. ДВОЙСТВЕННОСТЬ

Пары уравнений (14.14) и (14.15) или (14.14') и (14.15') иллюстрируют *двойственность*, существующую между точками и прямыми, задаваемыми на плоскости с однородными координата-

ми. Набор из трех чисел может представлять как прямую, так и точку, и, следовательно, уравнения прямой, заданной двумя точками, и уравнения точки, заданной пересечением двух прямых, идентичны. Принцип двойственности широко используется в проективной геометрии, поскольку для каждого свойства точек и прямых можно получить двойственный результат с помощью взаимной замены точки и прямой. Рассмотрим в качестве примера следующий результат.

Теорема **14.1.** Если прямые, соединяющие вершины (точки) двух треугольников, проходят через общую точку, то точки, в которых соответствующие стороны (прямые) пересекаются, лежат на общей прямой.

Эту теорему иллюстрирует рис. 14.2, а ее доказательство приведено в монографии [14.1]. Обратную теорему можно получить при помощи непосредственного использования принципа двойственности.

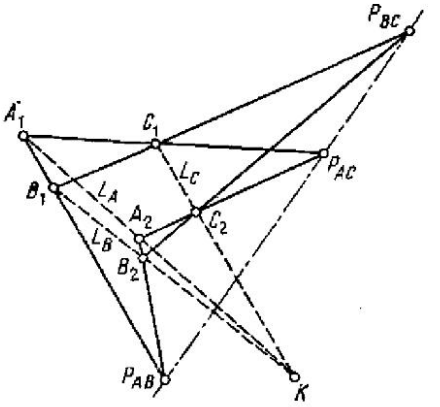


Рис. 14.2. Иллюстрация к теоремам 14.1 и 14.2. Точки P_{AB}, PAC и P_{BC} коллинеарны в том и только том случае, если прямые L_A, L_B и L_C проходят через общую точку

Теорема 14.2 (двойственная теореме 14.1). Если точки, в которых соответствующие стороны (прямые) двух треугольников пересекаются, лежат на общей прямой, то прямые, соединяющие соответствующие вершины (точки) треугольников, проходят через общую точку.

В машинной графике принцип двойственности можно использовать для экономной организации программного обеспечения. Так, одну и ту же процедуру можно применять как для определения пересечения двух прямых, так и для построения прямой, соединяющей две точки. (В обоих случаях такая процедура вычисляет значения миноров определителя размерами 3X3.) Однако читатель должен иметь в виду, что применять принцип двойственности можно лишь при использовании однородных координат. Более того, принцип двойственности применим лишь для получения результатов, относящихся к взаимному расположению объектов и не учитывающих никаких соотношений, связанных с расстоянием. Рассмотрим, например, следующий простой результат: «Точки

биссектрисы угла, образованного двумя прямыми, находятся на равном расстоянии от обеих прямых». В данном случае неясно, какой объект является двойственным биссектрисе, однако поскольку биссектриса представляет собой прямую, то двойственный объект должен быть соответственно точкой. Обозначим его через Q и сформулируем утверждение, двойственное только что приведенному: «Прямые, задаваемые двумя точками и проходящие через точку Q, находятся на равном расстоянии от этих двух точек». Поскольку существует лишь одна прямая, находящаяся на одинаковом расстоянии от двух точек (нормаль, проходящая через середину определяемого ими сегмента), то последнее заключенное в кавычки утверждение неверно [1].

14.4. ЗАДАЧИ, СВЯЗАННЫЕ С ОТРЕЗКАМИ ПРЯМЫХ

Решение многих задач, связанных с определением видности объектов изображения, так же как и задач отсечения, требует определения взаимного расположения прямых и точек. В данном разделе приводятся результаты, которые используются в последующих разделах и следующей главе. Во всех рассматриваемых случаях предполагается, что отрезки прямых определяются их концевыми упорядоченными точками. Если эти отрезки являются частями замкнутого контура, то упорядочение может выполняться как по часовой, так и против часовой стрелки. При отсутствии специальных указаний относительно упорядочения оно будет предполагаться следующим.

Допущение 14.1. Все отрезки прямых при отсутствии специальных оговорок ориентированы: у негоризонтальных отрезков стрелки направлены снизу вверх, у горизонтальных — слева направо. Другими словами, если (x_1, y_1, w_1) — координаты концевой точки, поставленной первой, и (x_2, y_2, w_2) — координаты концевой точки, поставленной второй, и если значение y_1 / w_1 не равно значению y_2 / w_2, то значение y_1 / w_1 меньше значения y_2 / w_2. В случае равенства (т. е. горизонтальности отрезка) значение x_1 / w_1 оказывается меньше значения x_2 / w_2 (рис. 14.3).

Определение 14.1. Мы будем считать, что точка P *расположена справа* от отрезка прямой L, если она расположена справа от наблюдателя, движущегося вдоль этой прямой от первой концевой точки ко второй (см. рис. 14.3).

Определение 14.2. Будем считать, что точка P *загораживает* прямую L (или отрезок прямой L'), если горизонтальная

[1] Мы не советуем читателю экспериментировать с определениями «биссектрисы», «эквидистанты» и т. п. с тем, чтобы попытаться сделать принцип двойственности применимым в этом примере. С большей пользой это время можно потратить на учебник по проективной геометрии с тем, чтобы лучше понять, почему принцип двойственности неприменим к соотношениям, содержащим метрику.

прямая, проходящая через точку P, пересекает прямую L (или отрезок L') в точке, значение координаты x которой меньше значения соответствующей координаты точки P (см. рис. 14.3).

Отношение «загораживает» не выполняется, если точка P лежит на прямой L, и не определено, если прямая L горизонтальна. Для отрезка прямой это отношение не определено в одном из сле-

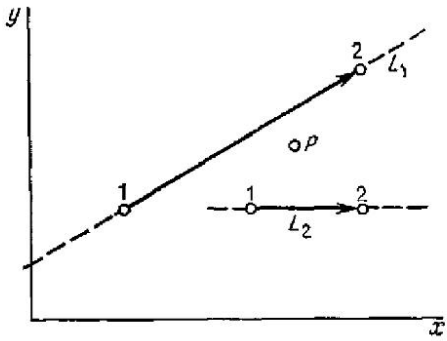


Рис. 14.3. Расшифровка обозначений, использованных в допущении 14.1 и определениях 14.1, 14.2: точка P расположена справа от прямой L_1 и слева от прямой L_2 ; кроме того, точка P загораживает прямую L_1

дующих трех случаев: точка P расположена выше верхней концевой точки отрезка, точка P расположена ниже нижней концевой точки отрезка прямой и точка P лежит на горизонтальном отрезке прямой. Если ориентация отрезка прямой соответствует допущению 14.1 и отношение «загораживает» определено, то оно становится эквивалентно отношению «лежит справа от».

14.4.1. ПОЛОЖЕНИЕ ТОЧКИ ОТНОСИТЕЛЬНО ПРЯМОЙ

Утверждение 14.1. Пусть X, Y и W — координаты точки P, а (x_1, y_1, w_1) и (x_2, y_2, w_2) — концевые точки отрезка прямой L. Если значения W, w_1 и w_2 положительны, то точка P расположена справа от прямой, определяемой отрезком прямой L, в том и только том случае, когда выполняется неравенство

$$X(y_1 w_2 - w_1 y_2) + Y(w_1 x_2 - x_1 w_2) + W(x_1 y_2 - y_1 x_2) < 0. \qquad (14.16)$$

Доказательство. Левая часть неравенства (14.16) равна значению определителя (14.14), равного 0, если точка P лежит на прямой, в которую входит отрезок прямой L. Поскольку определитель задает уравнение некоторой прямой, то точкам, расположенным по разные стороны этой прямой, соответствуют разные знаки этого определителя. Таким образом, для завершения доказательства необходимо лишь убедиться в том, что неравенство (14.16) выполняется для некоторой точки, расположенной справа от отрезка прямой L. Если прямая не горизонтальна, то можно выбрать точку с координатами

$$X = x_1 + w_1 \quad Y = y_1 \quad W = w_1.$$

Далее находим, что после деления на $w^2_1 w_2$ левая часть неравенства (14.16) принимает вид

$$\left(\frac{x_1}{w_1}+1\right)\left(\frac{y_1}{w_1}-\frac{y_2}{w_2}\right)-\frac{y_1}{w_1}\left(\frac{x_1}{w_1}-\frac{x_2}{w_2}\right)+\left(\frac{x_1 y_2}{w_1 w_2}-\frac{x_2 y_1}{w_2 w_1}\right).$$

Это выражение равно $y_1 / w_1 - y_2 / w_2$. Его значение отрицательно согласно допущению в порядке концевых точек. Если прямая горизонтальна, аналогичные преобразования выполняются для точки с координатами $X=x_1$ и $Y=y_1-w_1$.

При переходе от однородных координат к абсолютным утверждение 14.1 сохраняет силу, если принять $W=w_1 = w_2=1$. В частности, точка P расположена справа от прямой, если выполняется неравенство

$$X(y_1-y_2)+Y(x_2-x_1)+(x_1 y_2-y_1 x_2)<0. \qquad (14.16')$$

Воспользовавшись векторной записью (14.10), можно представить неравенства (14.16) и (14.16') в более компактной форме; кроме того, определитель матрицы 3X3 со столбцами a, b и c, будем обозначать как det(a, b, c). Преимущество такой записи заключается в том, что ее можно использовать и с абсолютными координатами, если приравнять третью координату всех векторов единице. В таком случае неравенство (14.16) [или (14.16')] принимает вид

$$\det(\mathbf{P}, \mathbf{P}_1, \mathbf{P}_2)<0. \qquad (14.16'')$$

Определения векторов P, P_1 и P_2 очевидны.

14.4.2. ПЕРЕСЕЧЕНИЕ ОТРЕЗКОВ ПРЯМЫХ

Приведенные выше результаты можно использовать для проверки наличия пересечения двух отрезков прямых и — в случае существования такого пересечения — для отыскания соответствующей точки. Если два отрезка прямых заданы точками P_1, P_2 и P_3, P_4, то они имеют пересечение в том и только том случае, когда при подстановке в уравнение прямой, соединяющей точки P_3 и P_4, координаты точек P_1 и P_2 имеют разные знаки. Аналогичное условие действует и для случая, когда точки P_1 ,P_2 и P_3, P_4 , меняются ролями (рис. 14.4). Итак, необходимо определять знаки следующих четырех величин:

$$S_1 = x_1(y_3 w_4 - w_3 y_4) + y_1(w_3 x_4 - y_3 w_4) + w_1(x_3 y_4 - y_3 x_4), \quad (14.17а)$$

$$S_2 = x_2(y_3 w_4 - w_3 y_4) + y_2(w_3 x_4 - x_3 w_4) + w_2(x_3 y_4 - y_3 x_4), \quad (14.17б)$$

$$S_3 = x_3(x_1 w_2 - w_1 y_2) + y_3(w_1 x_2 - x_1 w_2) + w_3(x_1 y_2 - y_1 x_2), \quad (14.17в)$$

$$S_4 = x_4(y_1 w_2 - w_1 y_2) + y_4(w_1 x_2 - x_1 w_2) + w_4(x_1 y_2 - y_1 x_2). \quad (14.17г)$$

Условие пересечения требует, чтобы S_1 и S_2 имели разные знаки; то же самое относится к 5з и *Si*. Отметим, что это условие несколько слабее условия, использованного в утверждении 14.1, по-

скольку в данном случае не важно, какой именно знак имеют соответствующие величины, если знаки различны.

Если условия пересечения выполнены, то координаты точки пересечения можно определить, решив пару линейных уравнений

$$x_1(y_1 w_2 - w_1 y_2) + y(w_1 x_2 - x_1 w_2) + w(x_1 y_2 - y_1 x_2) = 0, \quad (14.18а)$$

$$x(y_3 w_4 - w_3 y_4) + y(w_3 x_4 - x_3 w_4) + w(x_3 y_4 - y_3 x_4) = 0. \quad (14.18б)$$

Отметим, что числа уравнений и неизвестных одинаковы, поскольку значение w произвольно.

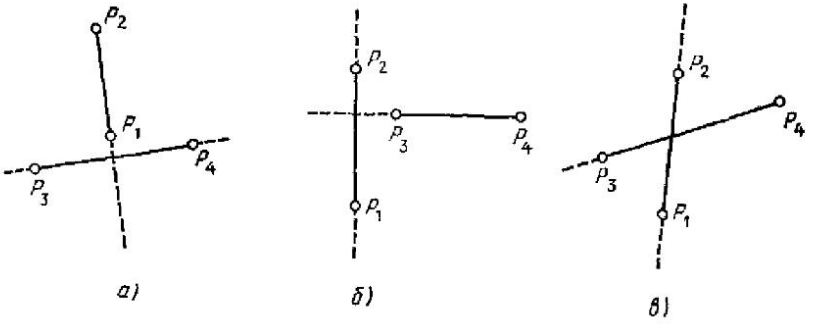


Рис. 14.4. Иллюстрация к соотношению знаков величин, определяемых уравнениями **(14.17)**:
а) $S_1S_2>0$ и $S_3S_4<0$; *б)* $S_1S_2<0$ и $S_3S_4>0$; *в)* $S_1S_2<0$ и $S_3S_4<0$

Условия пересечения можно записать в компактном виде:

$$S_1 = \det(\mathbf{P}_1, \mathbf{P}_3, \mathbf{P}_4), \quad S_2 = \det(\mathbf{P}_2, \mathbf{P}_3, \mathbf{P}_4), \quad S_1 S_2 < 0, \quad (14.19а)$$

$$S_3 = \det(\mathbf{P}_3, \mathbf{P}_1, \mathbf{P}_2), \quad S_4 = \det(\mathbf{P}_4, \mathbf{P}_1, \mathbf{P}_2), \quad S_3 S_4 < 0. \quad (14.19б)$$

Аналогичным образом уравнения (14.18) принимают вид:

$$\det(\mathbf{P}, \mathbf{P}_1, \mathbf{P}_2) = 0, \quad (14.20а)$$

$$\det(\mathbf{P}, \mathbf{P}_3, \mathbf{P}_4) = 0. \quad (14.20б)$$

Если одна или несколько величин S_i имеют нулевые значения, то имеет место вырожденный случай и критерий сравнения знаков использовать не удается[1]. Если $S_1 = 0$, то это означает, что точка P_1 лежит на прямой, определяемой точками P_3 и P_4. Чтобы определить ее принадлежность отрезку прямой, соединяющей эти точки, требуется лишь сравнить знаки S_3 и S_4. Если одно из этих значений равно 0, то это означает, что какие-то две точки совпадают. Если 0 равны оба этих значения, то все четыре рассмат-

[1] Если на евклидовой плоскости четыре точки выбираются случайным образом, то весьма маловероятно, что любые три из этих точек окажутся коллинеарны При переходе на дискретную плоскость, где значениями координат служат небольшие целые числа (например, от 0 до 511), ситуация меняется. В связи с ограниченностью точности воспроизведения изображений вырожденные случаи такого типа часто встречаются при решении прикладных задач.

риваемые точки коллинеарны. Наличие наложения отрезков прямых можно проверить с помощью сопоставления значений координат x или y.

14.4.3. ПОЛОЖЕНИЕ ТОЧКИ ОТНОСИТЕЛЬНО ПРЯМОУГОЛЬНИКА

Если вершины многоугольника упорядочены по часовой стрелке, то точка расположена внутри этого многоугольника, если она всегда находится справа от наблюдателя, совершающего обход сторон многоугольника в соответствии с порядком вершин. Если многоугольник выпуклый, то левая часть неравенства (14.16) имеет отрицательное значение для всех сторон многоугольника. В этом случае решение задачи очевидно.

Если многоугольник не является выпуклым, то найти решение уже не так просто. Можно показать, что любая точка, расположенная внутри невыпуклого многоугольника П, принадлежит выпуклому многоугольнику, образованному сторонами многоугольника П и их продолжениями ([3.7, p. 236—241]). Однако построение таких многоугольников представляет трудную задачу. При решении некоторых прикладных задач они могут задаваться заранее, как это бывает, например, в случаях, когда невыпуклый многоугольник построен как некоторое объединение выпуклых многоугольников. Затрата усилий на построение таких многоугольников может быть оправдана и в том случае, когда требуется проверка расположения относительно определенного невыпуклого многоугольника для значительного числа точек.

Другой способ решения, который применим к любому многоугольнику, предусматривает проведение через точку P прямой,

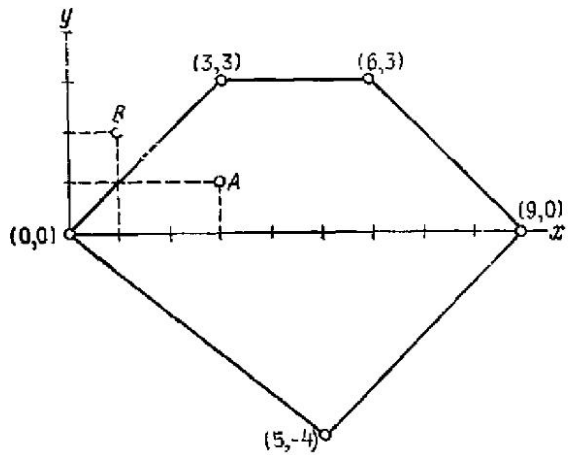


Рис. 14.5. Многоугольник, рассматриваемый в примере 14.4. Левые части первого (левого) набора неравенств для точки A с координатами (3.1) имеют значения - 6, - 6, - 15, - 28 и - 17. Левая часть первого неравенства для точки B с координатами (1,2) имеет значение 3

определение ее пересечения со всеми сторонами многоугольника П и использование критерия четности, описанного в разд. 8.1. Решение ряда систем уравнений более трудоемко, чем определение знаков системы неравенств. Кроме того, необходимо учесть затраты на сортировку точек пересечения. Таким образом, данный метод целесообразно применять лишь для работы с невыпуклыми многоугольниками. (Естественно, ситуация резко меняется, если требуется найти все точки, лежащие внутри многоугольника.)

Пример 14.4. Рассмотрим выпуклый многоугольник с вершинами (О, О), (3, 3), (6, 3), (9, 0) и (5, —4) (рис. 14.5). Пусть X и Y — координаты точки P. Данный случай определяется следующими неравенствами:

$$\begin{aligned} -3X+3Y &< 0 \quad \text{или } X>Y, \\ 3Y-9 &< 0 \quad \text{или } Y<3, \\ -3X+3Y-27 &< 0 \quad \text{или } X+Y<9, \\ 4X-4Y-36 &< 0 \quad \text{или } X-Y<9, \\ -4X-5Y &< 0 \quad \text{или } 4X>-5Y. \end{aligned}$$

14.4.4 ТЕНЬ ОТРЕЗКА

При решении ряда задач, связанных с разделением видимых и невидимых объектов изображения, требуется оценивать взаимное расположение отрезков прямых.

Определение 14.3. Будем считать, что отрезок прямой a затеняет или загораживает отрезок прямой b, если он не пересекает отрезок b и, по меньшей мере, одна из его точек загораживает отрезок b.

На рис. 14.6 приведено несколько примеров отрезков прямых и указано, какие отрезки затеняют другие в смысле приведенного определения. Отметим, что отношение «отрезок прямой а затеняет отрезок прямой b» нетранзитивно, так же как и его отрицание — отношение «отрезок прямой a не затеняет отрезок прямой b». Следовательно, отношение «затенение» не вводит частичное упорядочение, хотя в литературе можно встретить противоположные утверждения (см. разд. 14.5).

Приведем утверждение, представляющее собой непосредственное следствие результатов предыдущего раздела.

Утверждение 14.2. Отрезок прямой a затеняет отрезок пря-

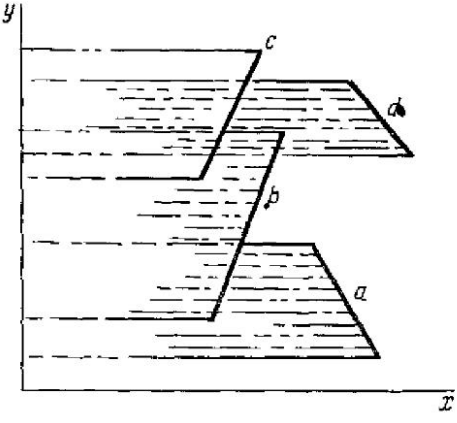


Рис. 14.6. Иллюстрация к затенению отрезка прямой: отрезок a затеняет отрезок b, отрезок b затеняет отрезок c, однако отрезок a не затеняет отрезок c, отрезок d затеняет отрезки b и с, однако не затеняет отрезок a

мой b в том и только в том случае, если выполняются следующие условия:

$$y_1^a \leqslant y_2^b, \tag{14.21а}$$

$$y_1^b \leqslant y_2^a, \tag{14.21б}$$

$$x_1^a (y_1^b - y_2^b) - y_1^a (x_1^b - x_2^b) + x_1^b y_2^b - y_1^b x_2^b < 0. \tag{14.21в}$$

14.5. БИБЛИОГРАФИЧЕСКИЕ ЗАМЕЧАНИЯ

Во многих упоминавшихся книгах по машинной графике обсуждается воспроизведение простых двухмерных изображений. Подробное рассмотрение однородных координат проводится в любом учебнике по проективной геометрии, например в [14.1, 14.4— 14.7]. Формулы преобразования координат при повороте можно найти в большинстве учебников по аналитической геометрии, например в [14.3]. Теоремы 14.1 и 14.2 принадлежат Дезаргу (1593— 1661 гг.). Предмет проективной геометрии — изучение геометрических свойств, являющихся инвариантными относительно линейных преобразований, например, типа тех, что реализуются с помощью умножения соответствующих матриц (см. разд. 14.2). Зарождение проективной геометрии относят к IV веку н. э. (Папп Александрийский) [1], однако зрелость она обретает лишь в XIX веке. Естественно, этот предмет заслуживает определенного внимания всех серьезно интересующихся машинной графикой. Обсуждение свойств отношения затенения и других аналогичных проблем можно найти в докладе [14.2].

Открытой проблемой машинной графики остается реализация некоторых геометрических преобразований на дискретной, а не на непрерывной плоскости. Ее разрешение могло бы привести к сокращению как объема вычислений, так и обмена между процессорами. Однако создается впечатление, что на этом пути нас ожидают непреодолимые трудности. Читатель может убедиться в том, что это действительно так, если попробует написать программу для осуществления поворотов на дискретной плоскости.

14.6. ЗАДАЧИ

14.1. Напишите программу для воспроизведения последовательно уменьшающихся изображений шестиугольника. Стягивание изображения должно продолжаться до тех пар, пака оно не превратится в один пиксел

14.2. Рассмотрим кривую C, заданную последовательностью точек $\{x_i, y_i\}^N$. Для воспроизведения такой кривой можно использовать последовательность команд *vec*, однако для того, чтобы воспроизведенная кривая имела определенную ширину, можно применить какое-либо «перо>. Рассмотрим случай пера с квадратным наконечником: воспроизведенная кривая будет представлять собой последовательность квадратов, центром каждого из которых служит точка (x_i, y_i), а одна из пар сторон параллельна прямой, задаваемой точками $(x_{i-1},$

[1] Согласно «Советскому энциклопедическому словарю» (3-е изд. — М. Сов. энциклопедия, 1985) Папп Александрийский жил в III веке н. э. — *Прим перев.*

y_{i-1}) и (x_i, y_i). Напишите программу, обеспечивающую воспроизведение такого изображения.

14.3. (Продолжение предыдущей задачи.) Квадратный наконечник не очень хорош с эстетической точки зрения. В качестве наконечника пера лучше использовать какой-либо эллипс. Напишите программу, обеспечивающую решение этой задачи с учетом того, что малая ось эллипса должна быть всегда параллельна вектору, соединяющему две последовательные точки отображения.

14.4. Модифицируйте программу-редактор, написанную в соответствии с заданием задачи 10.6, таким образом, чтобы в ней была предусмотрена возможность поворачивать, увеличивать и/или переносить кривые.

14.5. Напишите программу, обеспечивающую воспроизведение движения планет вокруг Солнца. Можно выбрать неравномерный масштаб для расстояния и временной интервал для обновления изображения, равный одной земной неделе. Используйте при решении задачи следующие упрощения: траекториями движения всех планет служат окружности, расположенные в одной и той же плоскости — это дает возможность применить уравнения, приведенные в разд. 14.2. Для обозначения каждой планеты можно выбрать различные многоугольники (скажем, квадрат — для Земли, шестиугольник — для Марса и т. п.), либо аналогичные многоугольники разных цветов, либо иной вариант в зависимости от того, калим типом дисплея Вы будете пользоваться

14.6. Напишите процедуру для определения пересечения двух прямых, представленных в однородных координатах, и используйте принцип двойственности для того, чтобы с помощью этой же процедуры можно было строить прямую, задаваемую двумя точками.

14.7. Напишите программу, в которой выполняется бесконечный цикл со следующими шагами:

1. Запрос у пользователя цветового кода и координат четырех точек. [(Можно предусмотреть использование цветового кода в качестве признака поступенного выхода из цикла.)
2. Воспроизведение отрезка прямой, соединяющего две заданные точки.
3. Задайте цвета всех воспроизведенных отрезков прямых таким образом, что при загораживании отрезком *a* отрезка *b* последний должен воспроизводиться в том же цвете, что и отрезок *a*. (Если Вы не можете воспользоваться цветным графическим дисплеем, помещайте рядом с изображением каждого отрезка какую-нибудь букву; эти буквы могут выступать в роли цветового кода.)

Глава 15 РАЗРЕЗАНИЕ МНОГОУГОЛЬНИКОВ

15.1. ВВЕДЕНИЕ

Термин «разрезание» используется для обозначения процесса установления того, пересекается ли прямая (либо многоугольник) с каким-либо многоугольником. Основная цель применения процедуры разрезания — выяснить, попадает ли некоторая совокупность объектов в определенное окно на экране дисплея. Следует подчеркнуть, что нельзя передавать функции разрезания дисплею по двум причинам. Во-первых, зона воспроизведения изображения может составлять лишь часть экрана дисплея (например, часть экрана может быть отведена для воспроизведения текста). Во-вторых, отсутствует формальный способ воспроизведения точек с недопустимыми координатами. Очень часто дисплей игнорирует старшие двоичные разряды, в результате чего изображение циклически переносится на экране с конца предыдущей строки в начало следующей. Таким образом, требуются алгоритмы, предназначенные для решения этой задачи.

Кроме того, разрезание используется для решения проблем видности. Чтобы определить, загораживает ли один объект другой, необходимо прежде установить, пересекаются ли они между собой. Третий способ применения процедуры разрезания, исследованный еще в очень незначительной степени, связан с разбиением невыпуклого многоугольника на выпуклые подмногоугольники. Такое разбиение может потребоваться не только в машинной графике (см. подразд. 14.4.3), но и в распознавании образов, где выпуклые подмножества многоугольника можно использовать в качестве непроизводных элементов для описания формы (очертаний) исходного многоугольника (см. разд. 15.7).

Результаты разд. 14.4 будут использованы для разработки ряда алгоритмов разрезания. В разд. 15.2 рассматривается разрезание отрезка прямой выпуклым многоугольником, а в разд. 15.3 дается решение этой задачи для одного практически важного случая: многоугольник представляет собой прямоугольник, стороны которого параллельны осям координат. Разрезанию произвольного многоугольника прямой посвящен разд. 15.4, а разрезанию многоугольника другим многоугольником — разд. 15.5. И, наконец, в разд. 15.6 исследуется проблема вычислительной эффективности решения некоторых геометрических задач.

15.2. РАЗРЕЗАНИЕ ПРОИЗВОЛЬНОГО ОТРЕЗКА ПРЯМОЙ ПРОИЗВОЛЬНЫМ ВЫПУКЛЫМ МНОГОУГОЛЬНИКОМ

Задача, обозначенная в названии данного раздела, характеризуется промежуточной степенью общности — крайние случаи соответствуют разрезанию произвольной прямой произвольным правильным прямоугольником (см. разд. 15.3) и разрезанию произвольной прямой произвольным невыпуклым многоугольником.

Задача. Если A_1 , A_2, ..., A_n — вершины выпуклого плоского многоугольника П, требуется определить, какая часть отрезка прямой L (заданного концевыми точками P_1 и P_2) лежит внутри этого многоугольника.

Заметим, что «очевидное» решение, предусматривающее проверку принадлежности точек P_1 и P_2 внутренней части многоугольника, неприемлемо, поскольку отрезок прямой может пересекать выпуклый многоугольник даже в том случае, если обе концевые точки отрезка лежат за пределами многоугольника. Допустимым является следующее решение, в основе которого лежат результаты разд. 14.4.

Вначале вычисляются n значений величины

$$S_i = \det(\mathbf{A}_i, \mathbf{P}_1, \mathbf{P}_2) = X_i(y_1 w_2 - w_1 y_2) + Y_i(w_1 x_2 - x_1 w_2) + W_i(x_1 y_2 - y_1 x_2),\ i = 1, 2, \ldots, n, \quad (15.1)$$

где A_i — вектор координат (X_i, Y_i) вершины A_i многоугольника и P_j $(j=1, 2)$ — вектор координат (x_i, y_j) концевых точек прямой P_j. Полученные результаты могут соответствовать одному из следующих случаев.

1. Все значения S_i — ненулевые и имеют один и тот же знак. Следовательно, все вершины многоугольника расположены по од ну сторону отрезка прямой L и никакая часть прямой не заключе на внутри многоугольника.

2. Одно из значений S_i равно 0, а все остальные значения име ют один и тот же знак. Следовательно, прямая проходит через одну из вершин многоугольника, а все остальные его вершины расположены по одну сторону этой прямой. Данный случай эк вивалентен случаю 1.

3. Два значения S_i равны 0, а все остальные значения имеют один и тот же знак. Поскольку многоугольник П — выпуклый, это может означать лишь то, что две соседние вершины много угольника лежат на отрезке прямой L. Следовательно, это — вы рожденный случай, рассмотренный в подразд. 14.4.2: одна из сто рон многоугольника расположена на той же прямой, что и отрезок прямой L.

4. Одно или два значения S_i равны 0, а остальные значения имеют разные знаки. Следовательно, отрезок прямой L пересека ет многоугольник П, проходя через одну или две его вершины. Поскольку многоугольник П — выпуклый, знаки изменяются лишь дважды. Этот случай будет рассматриваться вместе со следую щим.

5. Все Si имеют ненулевые значения и знаки их различны. Это означает, что при обходе многоугольника по периметру знак из меняется дважды. Для обозначения пар вершин, на которых про исходит перемена знака, будут использоваться символы A_j, A_{j+1} и A_k, A_{k+1}. В случае 4 величина Si может принимать ненулевое значение в одной из вершин одной или обеих пар. Отрезок пря мой, соединяющий вершины первой пары, обозначается через L_1, а отрезок прямой, соединяющий вершины второй пары, — через L_2. Отметим, что не имеет значения, какая пара назначается пер вой, а какая — второй; то же самое относится к соответствующим обозначениям, если принятая процедура обработки точно выпол няется.

После установления пересечения прямой, содержащей отрезок L, рассматриваемого многоугольника (два последних случая), не-обходимо определить расположение точек L_1 и L_2 относительно отрезков прямых L_1 и L_2, т. е. знаки следующих четырех величин:

$$U_1 = \det(\mathbf{P}_1, \mathbf{A}_j, \mathbf{A}_{j+1}) = x_1(Y_j W_{j+1} - W_j Y_{j+1}) + y_1(W_j X_{j+1} - X_j W_{j+1}) + w_1(X_j Y_{j+1} - Y_j X_{j+1}) \quad (15.2a)$$

(расположение точки P_1 относительно отрезка прямой L_1);

$$U_2 = \det(\mathbf{P}_2, \mathbf{A}_j, \mathbf{A}_{j+1}) = x_2 (Y_j W_{j+1} - W_j Y_{j+1}) + y_2 (W_j X_{j+1} - X_j W_{j+1}) + w_2 (X_j Y_{j+1} - Y_j X_{j+1}) \quad (15.2б)$$

(расположение точки P_2 относительно отрезка прямой L_1);

$$U_3 = \det(\mathbf{P}_1, \mathbf{A}_k, \mathbf{A}_{k+1}) = x_1 (Y_k W_{k+1} - W_k Y_{k+1}) + y_1 (W_k X_{k+1} - X_k W_{k+1}) + w_1 (X_k Y_{k+1} - Y_k X_{k+1}) \quad (15.2в)$$

(расположение точки P_1 относительно отрезка прямой L_2);

$$U_4 = \det(\mathbf{P}_2, \mathbf{A}_h, \mathbf{A}_{h+1}) = x_2 (Y_h W_{h+1} - W_h Y_{h+1}) + y_2 (W_h X_{h+1} - X_h W_{h+1}) + w_2 (X_h Y_{h+1} - Y_h X_{h+1}) \quad (15.2г)$$

(расположение точки P_2 относительно отрезка прямой L_2).

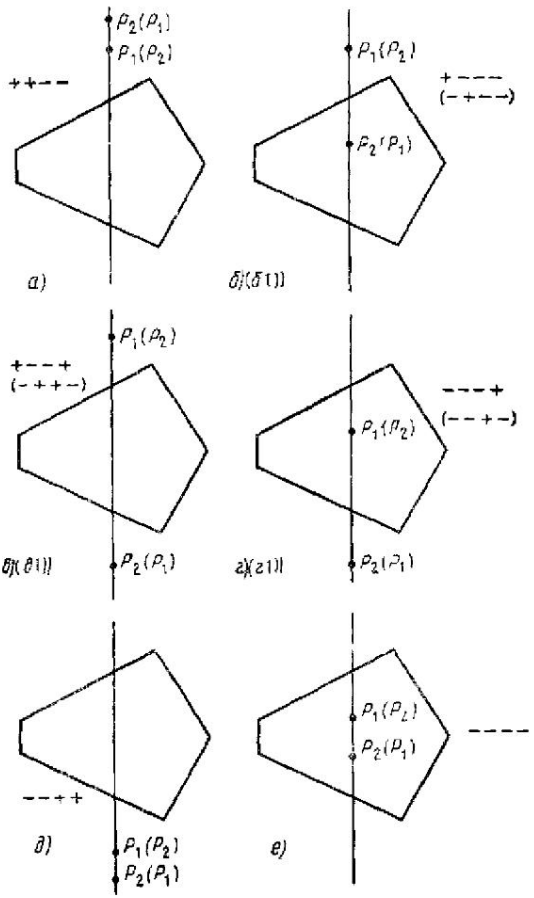


Рис. 15.1. Шесть возможных вариантов разрезания произвольного отрезка прямой произвольным многоугольником

При упорядочении ориентации сторон многоугольника по часовой стрелке величина U_i имеет в уравнении (15.2) отрицательный знак, если точка лежит внутри многоугольника Значение U_i равно 0, если одна из точек P_1 и P_2 лежит на одной из сторон многоугольника. Мы не будем рассматривать здесь этот вырожденный случай и предлагаем провести соответствующую модификацию алгоритма 15.1 в качестве упражнения (см. задачу 15.1). На рис. 15.1 приведены различные варианты расположения точек P_1 и P_2 относительно двух прямых, а табл. 15.1 содержит знаки U_i и результаты определения концевых точек воспроизводимого отрезка. В табл. 15 1 и алгоритме 15.1 используются следующие обозначения: Q_1 и Q_2 — концевые точки отрезка прямой, изображение которого должно быть воспроизведено; $L \bullet L_i$ (i=l, 2) — точка пересечения отрезков L и L_i.

Таблица 15.1

Расположение произвольного отрезка прямой относительно произвольного многоугольника

Рис 15 1	U_1	U_2	U_3	U_4	Q_1	Q_2
а)	+	+	—	—	невидима	
б)	+	—	—	—	$L \cap L_1$	P_2
б1)	—	+	—	—	P_1	$L \cap L_1$
в)	+	—	—	+	$L \cap L_1$	$L \cap L_2$
в1)	—	+	+	—	$L \cap L_2$	$L \cap L_1$
г)	—	—	—	+	P_1	$L \cap L_2$
г1)	—	—	+	—	$L \cap L_2$	P_2
д)	—	—	+	+	невидима	
е)	—	—	—	—	P_1	P_2

Теоретически имеется 16 невырожденных вариантов (число различных наборов знаков у четырех переменных), однако в силу геометрических ограничений в действительности возможны лишь 9 случаев. В самом деле, невозможно, чтобы все значения U_i были положительными, так как положительные полуплоскости для двух сторон расположены вне многоугольника. Аналогичным образом, невозможно, чтобы три U_i имели положительные знаки. В результате остается 11 вариантов. Конфигурация « - + - + » означает, что точка $P1$ находится внутри многоугольника, а точка P_2 расположена выше прямой A_jA_{j+1} и ниже прямой A_kA_{k+1}, что невозможно. Аналогичным образом можно интерпретировать конфигурацию « + - + - ». В результате остается 9 вариантов, сведенных в табл. 15.1. Отметим, что, обращаясь к реальным геометрическим построениям, мы обнаруживаем лишь 6 случаев, которые представлены на рис. 15.1. Три из них можно маркировать двумя допустимыми способами — это случаи, представленные на рис. 15.1,*б-г* (второй вариант маркировки дан в скобках). Если допустить, что точка $P1$ расположена ниже точки P_2, то можно ог-

раничиться рассмотрением лишь 6 случаев, однако это допущение не следует принимать, когда P_1P_2 является стороной многоугольника и упорядочение концевых точек не определяется значениями их координаты y.

Алгоритм 15.1 определяет знаки величин, задаваемых уравнениями (15.1) и (15.2), и использует решения, перечисленные в табл. 15.1. Первая часть алгоритма обеспечивает определение расположения прямой относительно вершин прямоугольника, а вторая часть — определение расположения концевых точек прямой. Переменным v и u присваиваются значения, соответствующие знакам соседних вершин многочлена S_i, которые находятся из уравнения (15.1). Если значения u и v неодинаковые, то значит, две соответствующие вершины расположены с противоположных сторон прямой, содержащей отрезок L. Координаты таких вершин запоминаются на шаге 5, K — счетчик числа таких пересечений. Поскольку предполагается, что многоугольник выпуклый, их число может быть равно 0 или 2. Если по окончании цикла, включающего шаги 3—6, оказывается, что это число равно 0, то это означает отсутствие пересечения прямой, содержащей отрезок L, и многоугольника, и выполнение алгоритма прекращается (шаг 7). В противном случае осуществляются шаги 8—17.

На шаге 9 вычисляются значения величин, определяемых уравнениями (15.2), и на остальных шагах выполняются тесты, предусмотренные табл. 15.1. Случаи а) и д) — отрезок прямой лежит вне многоугольника — исследуются на шаге 10. Если условие, проверяемое на шаге 10, не выполняется, то обратившись к рис. 15 1 и табл. 15.1, можно установить, что при противоположных знаках U_1 и U_3 прямая L пересекает один из отрезков L_1 и L_2. Аналогичным образом интерпретируется противоположность знаков U_2 и U_4. Проверка этих условий проводится на шагах 14 и 15: сначала для U_1 и U_3, затем для U_2 и U_4.

Исходя из принципа двойственности, алгоритм разрезания можно использовать и для решения следующей задачи.

Задача. Задано n прямых, являющихся сторонами выпуклого многоугольника. Требуется определить, лежит ли внутри этого многоугольника точка, заданная пересечением двух прямых.

Решение. Примените алгоритм, двойственный алгоритму 15.1.

Алгоритм 15.1. Разрезание произвольного отрезка прямой произвольным выпуклым многоугольником

Обозначения. Массив A имеет размерность $nx3$ и содержит тройки значений однородных координат n вершин многоугольника. Координаты концевых точек отрезка прямой L хранятся в массиве P размерностью 2x2. Размерности массивов B и С равны 2X3. Массив B используется для хранения первой из пары вершин, расположенных по разные стороны прямой, содержащей отрезок L; массив C содержит вторую вершину таких пар. S_i и U_i определяются уравнениями (15.1), (15.2).

Алгоритм определяет Q_1 и Q_2 — концевые точки отрезка, расположенного внутри многоугольника.

1. Значение счетчика K приравнивается 0.
2. Переменной v присваивается значение, соответствующее зна ку S_1, который определяется из уравнения (15.1) при исполь зовании A (1,1) в качестве X\, A (1,2) — в качестве Y_1 и т. д.
3. For $i=2$ до n do:
   Begin.
4. Переменной u присваивается значение, соответствующее знаку S_i, который определяется из уравнения (15.1) при использовании A $(i,$ 1) в качестве X_i, A $(i,2)$ — в каче стве Уг И Т. Д.
5. If v не равно «, then значение счетчика K увеличивается, B (K, j) присваивается значение A $(i—1, j)$, C (K, j) — A (i, j) и задается $v = u$.
6. If значение счетчика K равно 2, then break из цикла.
   End.
7. If значение счетчика K равно 0, then exit.
8. Else do:
   Begin.
9. Определяются уравнения (15.2): из массива Б (1,*) бе рется точка A_j, из массива C (1, *) — A_{j+1}, из массива B (2, *) — A_k, и из массива C (2, *) — A_{k+1}. После это го вычисляются значения U_1—U_4.
10. If U_1 и U_3 имеют одинаковые знаки и этот знак противо положен знаку при [U_2 и U_4, then exit.
11. Else do:
    Begin.
12. For m = 1 до 2 do:
    Begin.
13. If U_m и U_{m+2} имеют противоположные знаки, then do:
    Begin.
14. If U_m имеет *отрицательное* значение, then Q_m квалифицируется как точка пересечения отрезков прямых L и L_2.
15. Else Qm квалифицируется как точка пересечения отрезков прямых L и L_1.
    End.
16. Else в качестве Q_m используется точка P_m.
    End.
    End.
    End.
17. Воспроизводится отрезок прямой, соединяющий точки Q_1 и Q_2
18. Конец алгоритма.

15.3. РАЗРЕЗАНИЕ ПРОИЗВОЛЬНОГО ОТРЕЗКА ПРЯМОЙ ПРОИЗВОЛЬНЫМ ПРАВИЛЬНЫМ ПРЯМОУГОЛЬНИКОМ

Рассмотрим частный случай алгоритма разрезания. Он возникает при $n=4$, когда соответствующий многоугольник представляет собой прямоугольник со сторонами, параллельными осям координат. Будем называть такой прямоугольник *правильным* [1]. Существуют достаточно простые способы определения расположения точек P_1 и P_2 относительно вертикальных и горизонтальных прямых, ограничивающих прямоугольник. На рис. 15.2 участок 5 характеризует наблюдаемый фрагмент изображения (окно наблюдения). Если точки P_1 и P_2 находятся на одном из участков, за исключением участка 5, либо на участках одной из групп (1, 4, 7), (1, 2, 3), (3, 6, 9) или (7, 8, 9), то отрезок не виден. Он становится виден, когда обе точки находятся на участке 5. Проблему видности отрезка следует тщательно изучить для всех вариантов расположения концевых точек.

Этот частный случай столь важен с практической точки зрения, что для него целесообразно написать специальный алгоритм разрезания — здесь он приводится в виде алгоритма 15.2. На шаге 1 проверяется, не лежит ли рассматриваемый сегмент целиком в одной из четырех групп, каждая из которых включает по три участка, как показано на рис. 15.2: (1, 4, 7), (3, 6, 9), (7, 8, 9), (1, 2, 3). Если это так, то отрезок находится за пределами наблюдаемого фрагмента изображения и, следовательно, выполнение алгоритма можно прекратить. На шаге 2 проверяется соответствие разметки концевых точек допущению 14.1. Шаг 3 предназначен для обработки горизонтальной прямой (с учетом шагов 3а—3в), а шаг 4 (с учетом шагов 4а и 4*б*) — для обработки вертикальной прямой. Шаги 5—11 предназначены для анализа общего случая. Шаги 1, 3 и 4 без всякого ущерба можно было опустить, однако это приведет к увеличению объема вычислений. Эти шаги позволяют быстро проанализировать случаи, в которых не требуется использования общего критерия пересечения. Таким образом, это пример ситуации, когда увеличение длины алгоритма приводит к уменьшению среднего времени его выполнения.

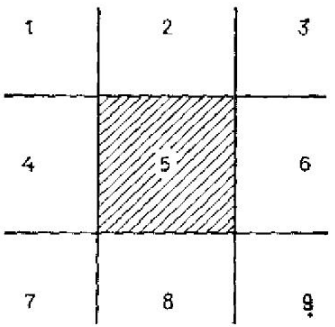


Рис. 152. Разрезание при помощи прямоугольника окна наблюдения

На шаге 5 определяется знак четырех вершин относительно рассматриваемой прямой. В частности, U_1 соответствует нижней

[1] В принципе, правильным принято называть плоский выпуклый многоугольник, у которого равны все стороны и все углы при вершинах. — *Прим. перге.*

левой вершине (X_{MIN}, Y_{MIN}), U_2 — вершине (X_{MIN}, Y_{MAX}), U_3 — верхней правой вершине (X_{MAX}, Y_{MAX}) иU_4 — вершине (X_{MAX}, Y_{MIN}). Условие, проверяемое на шаге 7, выполняется, если прямая пересекает левую границу наблюдаемого фрагмента изображения. На шагах 7*б* и 7*в* определяются координаты точки пересечения, которые во всех последующих вычислениях используются в качестве координат концевой точки отрезка. Отметим, что условия, проверяемые на этих двух шагах, не могут выполняться одновременно. Если бы эти условия выполнялись одновременно (т. е. x_1 и x_2 были бы меньше X_{MIN}), то алгоритм должен был бы закончиться после выполнения шага 1. Условие, проверяемое на шаге 8, выполняется, если прямая пересекает верхнюю границу наблюдаемого фрагмента изображения. На шаге 8*б* определяются координаты точки пересечения. Отметим, что в силу допущения 14.1 необходимо оценивать расположение только одной концевой точки. Условие, проверяемое на шаге 9, выполняется, если прямая пересекает правую границу наблюдаемого фрагмента изображения, а условие, проверяемое на шаге 10, выполняется, если прямая пересекает нижнюю границу наблюдаемого фрагмента изображения. На шагах 9*б*, 9*в* и 10*б* определяются координаты точек пересечения, которые затем используются в качестве координат концевых точек.

Отметим, что лишь два из четырех условий, проверяемых на шагах 7—10, могут выполняться. Следовательно, выполнение алгоритма можно ускорить, прекратив применять тест знака после того, как j приняло значение 2. Кроме того, следует отметить, что когда отрезок прямой целиком попадает в наблюдаемый фрагмент изображения, ни одно из условий 7*б*, 7*в*, 8*б*, 9*б*, 9*в* и 10*б* не может выполняться и единственной нетривиальной операцией является шаг 5.

Алгоритм 15.2. Разрезание произвольного отрезка прямой произвольным правильным прямоугольником

Обозначения. Концевые точки прямой имеют координаты (x_1, y_1) и (x_2, y_2). Прямоугольник определяется координатами X_{MIN} и X_{MAX} , Y_{MIN} и Y_{MAX}. Процедура *draw* (вычерчивание) обеспечивает воспроизведение изображения прямой. *dy, dx, dxy, qx, qX, qy* и *qY* — вспомогательные величины, определенные на шаге 5.

1. If (x_1 и x_2 меньше X_{MIN}), или (x_1 и x_2 больше X_{MAX}), или (y_1 и y_2 меньше Y_{MIN}), или (y_1 и y_2 больше Y_{MAX}), **then return.**
2. If y_1 больше y_2, **then** концевые точки меняются местами.
3. If y_1 равно y_2, **then.**
   Begin.

3*а*. If x_1 больше x_2, **then** концевые точки меняются местами.

3*б*. If x_1 меньше X_{MIN}, **then** задается $x_1 = X_{MIN}$. If x_2 больше X_{MAX}, **then** задается $x_2 = X_{MAX}$.

3в. *draw* (x_1, y_1, x_2, y_2) и return.
End.

4. If x_1 равно x_2, then.
Begin.

4а. If y_1 меньше Y_{MIN}, then задается $y_1 = Y_{MIN}$.. If y_2 больше Y_{MAX}, then задается y_2—Y_{MAX}.

4б. *draw* (x_1, y_1, x_2, y_2) и return.
End.

5. Задается

$dy = y_1—y_2$; $dx = x_1—x_2$; $dxy = x_1y_2 - y_1x_2$. $qx = X_{MIN}dy$; $qX = X_{MAX}dy$; $qy = Y_{MIN}dx$; $qY = Y_{MAX}dx$. $U_1 = qx—qy + dxy$, $U_2 = qx—qY + dxy$, $U_3 = qX—qY + dxy$, $U_4 = qX—qy + dxy$

6. Величине j присваивается О.

7. If U_1 и U_2 имеют разные знаки, then.
Begin.

7а. Производится приращение j.

7б. If x_1 меньше X_{MIN}, then задается

$y_1 = (X_{MIN}\,dy + dxy)/dx$; $x_1 = X_{MIN}$.

Производится коррекция dx, dy и dxy.

7в. If x_2 меньше X_{MIN}, then задается

$y_2 = (X_{MIN}\,dy + dxy)/dx$; $x_2 = X_{MIN}$.

Производится коррекция dx, dy и dxy. End.

8. If U_2 и U_3 имеют разные знаки, then
Begin.

8а. Производится приращение j.

8б. If y_2 больше Y_{MAX} then

$x_2 = (Y_{MAX}\,dx—dxy)/dy$; $y_2 = Y_{MAX}$.

Производится коррекция dx, dy и dxy. End.

9. If U_3 и U_4 имеют разные знаки, **then**
Begin.

9а. Производится приращение j.

9б. **If** x_1 больше X_{MAX}, then задается

$y_1 = (X_{MAX}\,dy + dxy)/dx$; $x_1 = X_{MAX}$.

Производится коррекция dx, dy и dxy.

9в. **If** x_2 больше X_{MAX}, **then** задается

$y_2 = (X_{MAX}\,dy + dxy)/dx$; $x_2 = X_{MAX}$.

Производится коррекция dx, dy и dxy. **End.**

0. **If** U_4 и U_1 имеют разные знаки, then **Begin.**

10а. Производится приращение j.

10б. If r/i меньше Y_{MIN}, then задается

$x_1 = (Y_{MIN}\, dx—dxy)/dy$; $y_1 = Y_{MIN}$.

Производится коррекция dx, dy и dxy.
End.

11. If j больше O, then *draw* (x_1, y_1, x_2, y_2).
12. Return.
13. Конец алгоритма.

15.4. РАЗРЕЗАНИЕ ПРОИЗВОЛЬНОГО МНОГОУГОЛЬНИКА ПРОИЗВОЛЬНОЙ ПРЯМОЙ

При воспроизведении изображения многоугольника, который может не быть выпуклым на участке окна наблюдения, представляющего собой правильный прямоугольник, можно либо рассмотреть разрезание каждой из сторон многоугольника окном (с помощью алгоритма 15.2), либо разрезание всего многоугольника каждой из сторон окна. Для решения последней задачи предназначен алгоритм 15.3. Его можно также использовать в качестве базового алгоритма для определения пересечения двух произвольных многоугольников.

Этот алгоритм предусматривает вычисление U_1 и U_2 для всех вершин с помощью уравнений (15.2). (Поскольку в процессе работы алгоритма пары вершин обрабатываются последовательно, новое значение U_2 равно предыдущему значению U_1 и, следовательно, U для каждой вершины должно вычисляться только один раз.) Предполагается, что направление прямой соответствует допущению 14.1 и видны те точки, которые не загораживают прямую (см. определение 14.2). Таким образом, если значение U_i, положительно, то соответствующая вершина видна. На шаге 3 сравниваются знаки U_1 и U_2 и, если хотя бы одно из этих значений положительно, алгоритм переходит к выполнению шагов 4—7. На шаге 5 определяется пересечение стороны многоугольника с прямой.

Теоретически невозможно, чтобы на шаге 5 оказалось, что значение w_a равно 0, поскольку это означает параллельность отрезка и прямой, т. е. одинаковость знаков U_1 и U_2 и, следовательно, возникновение противоречия В табл. 152 дана сводка шагов алгоритма, выполняемых в зависимости от соотношения знаков U_1 и U_2.

Если обе точки лежат на прямой, алгоритм классифицирует отрезок как невидимый. Фактически, воспроизводимый многоугольник оказывается разомкнутым там, где прямая пересекает его, как показано на рис. 15.3. При решении некоторых прикладных задач такое положение может быть нежелательным, однако описываемый алгоритм нетрудно изменить таким образом, что точки пересечения будут соединяться прямыми (см. задачу 15.4).

Таблица 15.2

Положение многоугольника относительно прямой

U_1	U_2	Статус видности отрезка	Шаг алгоритма
+	+	полностью виден	4
+	0	полностью виден	4
+	—	частично виден	7
0	+	полностью виден	4
0	0	не виден	3
0	—	частично виден	7
—	+	частично виден	6
—	0	не виден	3
—	—	не виден	3

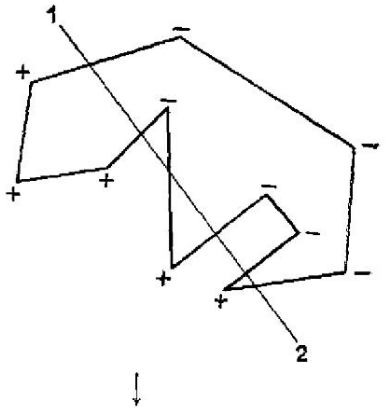


Рис. 15 3. Пример многоугольника, разрезанного прямой. Каждая вершина многоугольника помечена знаком величины U_i, вычисленной для этой вершины

Алгоритм 15.3. Разрезание произвольного прямоугольника произвольной прямой

Обозначения. В качестве исходных данных алгоритму задаются коэффициенты уравнения прямой *dY, dX* и *D:*

$$dYx - dXy + D = 0$$

и, кроме того, задаются координаты вершин многоугольника x_i, y_i, $i=1, 2, \ldots, N$. В результате должно быть воспроизведено порезанное изображение многоугольника, который не загораживается прямой *dy, dx* и *dd* — коэффициенты прямой, определяемой стороной многоугольника, которые вычисляются на шаге 5 Однородные координаты точки пересечения определяются методом, изложенным в подразд. 14.3.2.

1. Задается $U_i = dYx_1 — dXy_2 + D$.
2. For *i om 2* до *N* do шаги 3—8.

Begin.

3. Задается $U_2 = dYx_1 — dXy_2 + D$.
If U_1 и U_2 неположительны, **then do** ничего.
Else

Begin.

4. If U_1 и U_2 неотрицательны, **then** *draw* вектор из ТОЧКИ (x_{i-1}, y_{i-1}) В ТОЧКУ (x_i, y_i).
Else

Begin.

5. Задается

$$dy = y_{i-1} - y_i,\ dx = x_{i-1} - x_i$$

$$dd = x_{i-1} y_i — y_{i-1} x_i,$$

далее вычисляются $x_a =$ $—(dxD—dXD)$,

$y_a = — (dyD—dYD)$,

$w_a = — (dydX—dYdx)$.

If w_a имеет ненулевое значение, **then** задается $x_{пер} = x_a/w_a$ И $y_{пер} = y_a/w_a$.

Else выводится признак ошибки и процесс заканчивается.

6. If U_1 отрицательно, **then** *draw* вектор из ТОЧКИ $(x_{пер}, y_{пер})$ в точку (x_i, y_i).
   Else
   Begin.
7. {U_1 неотрицательно и U_2 положительно}
   Draw вектор из точки (x_{i-1}, y_{i-1}) в точку $(x_{пер}, y_{пер})$
   End. End.
   End.
8. Задается $U_1 = U_2$.
   End.
9. Конец алгоритма.

15.5. ПЕРЕСЕЧЕНИЕ ДВУХ МНОГОУГОЛЬНИКОВ

Определение пересечения двух многоугольников представляет существенный интерес для решения проблемы разделения видимых и невидимых элементов воспроизводимого объекта и является наиболее общей постановкой задачи разрезания. Задачу определения пересечения двух многоугольников можно решить при помощи многократного применения алгоритма 15.3. Рассмотрим многоугольники $П_1$ и $П_2$, причем будем считать, что многоугольник $П_2$ — выпуклый. Возможно следующее решение поставленной задачи. Пусть $q_1, q_2, ..., q_n$ — стороны многоугольника $П_2$ и P_i, $i = = 1, 2, ..., т$, — вершины многоугольника $П_1$, который может быть как выпуклым, так и невыпуклым. Воспользуемся алгоритмом 15.3 для определения разрезания многоугольника $П_1$ каждой стороной q_i многоугольника $П_2$. В результате разрезания $П_1$ стороной q_1 возникнут новые многоугольники, число которых может изменяться от 0 до увеличенного на 1 числа вогнутых вершин $П_1$. Поскольку число получающихся многоугольников может оказываться довольно значительным, при решении этой задачи должен быть налажен тщательнейший учет[1].

[1] Внимательный читатель мог бы возразить против использования этого метода, указав, что алгоритм 15.3 предназначен для разрезания произвольного многоугольника произвольной прямой, а не произвольным отрезком прямой. Допустимость такой подмены, однако, следует из введенного нами допущения о том, что многоугольник $П_2$ — выпуклый.

Для описания $П_1$ и всех остальных многоугольников, которые могут возникнуть в процессе разрезания, лучше всего использовать связный список. В частности, каждой вершине ставится в соответствие указатель, который при обходе многочлена по часовой стрелке указывает следующую вершину. Для обозначения указателя будет применяться символ $NxT(P_i)$. Вначале $NxT(P_i) = P_{i+1}$ при $i<m$ и $NxT(P_m)=P_1$. Перейдем теперь к модификации алгоритма 15.3. Вместо проведения сторон многоугольников будут просто корректироваться соответствующие указатели. В частности, на шаге 6 в список вводится точка пересечения и устанавливается $NxT(P_{m+j})=P_i$, где j — число, обозначающее позицию новой точки в конце списка. На шаге 7 устанавливается $NxT(P_{i-1}) = P_{m+j}$ и $NxT(P_{m+j})=0$, поскольку следующая точка пока не известна. Если вершина невидима, то ее указателю также присваивается значение 0. Элементы этой процедуры иллюстрируются на рис. 15.4, где стрелками отмечены результаты использования указателей. Для придания процедуре законченности необходимо задать указатели для связывания точек пересечения. Для этого точки P_{m+j} сортируются в порядке увеличения значений координаты y или x, если режущая прямая горизонтальна. Затем эти точки берутся попарно и указателем с нулевыми значениями присваиваются значения адресов вторых элементов соответствующей пары. Так, в случае, приведенном на рис. 15.4, вводятся две связи:

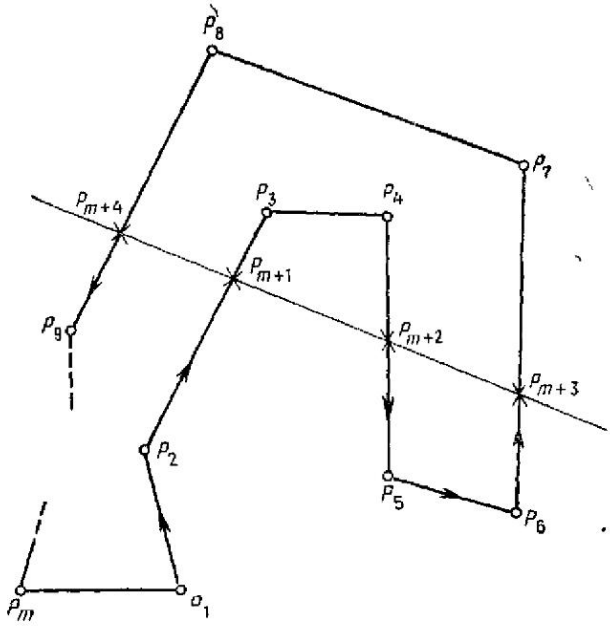


Рис. 15.4. Иллюстрация к расстановке указателей при разрезании произвольного многоугольника произвольной прямой

$NxT\ (P_{m+1}) = P_{m+4}$, $NxT\ (P_{m+3}) = P_{m+2}$.

Теперь совсем нетрудно обойти описок вершин и найти новые многоугольники. На самом деле, эту процедуру не требуется доводить до конца, поскольку при использовании алгоритма 15.3 фактически не имеет значения число многоугольников. Вероятно, наилучшим способом обхода списка вершин является использование еще одного набора указателей, содержащих адрес в списке одной точки каждого многоугольника. Этот набор следует просматривать последовательно, используя каждый очередной указатель как предписание полного обхода контура соответствующего многоугольника.

Описанная выше процедура предназначена для определения пересечения двух многоугольников посредством нахождения пересечений многоугольника $П_1$ полуплоскостями H_i, заданными сторонами многоугольника $П_2$. (Прямая, содержащая сторону q_i, делит плоскость на две части. Полуплоскость H_i представляет собой ту часть плоскости, которая содержит точки, расположенные внутри многоугольника в окрестности стороны q_i.) Если $П_2$ — невыпуклый, то этот метод приводит к определению пересечения $П_1$ не с $П_2$, а с пересечением полуплоскостей $N = • H_i$. Последнее *(N)* называют *центром* или *ядром* многоугольника $П_2$. Оно представляет собой выпуклый многоугольник, обладающий тем свойством, что прямая, проведенная из любой его точки в любую вершину многоугольника $П_2$, полностью лежит в последнем. (Естественно, может оказаться, что ядро является пустым множеством.) Рассмотрение ядер и их свойств выходит за пределы задач нашей книги (см. разд. 15.7).

С другой стороны, алгоритм 15.3 можно использовать для разбиения невыпуклого многоугольника на выпуклые. Рассмотрим, в частности, многоугольник П, один из вогнутых углов которого образован сторонами q_j и q_{i+1}. Разрезание многоугольника $П_2$ сначала стороной q_j , а затем стороной q_{j+1} приводит к возникновению двух многоугольников: $П• H_j$ и $П• H_{j+1}$. В каждом из них число вогнутых углов хотя бы на единицу меньше, чем в многоугольнике П. Повторив эту процедуру для всех сторон, образующих вогнутые углы, получим набор многочленов, объединение которых равно П. Эти многоугольники можно разрезать, используя стороны, образующие их вогнутые углы, и так далее, вплоть до тех пор, пока все полученные многочлены не окажутся выпуклыми (см. задачи 15.6, 15.7).

Вместо разбиения многоугольника $П_2$ на выпуклые многоугольники можно следующим образом изменить способ использования алгоритма 15.3. Каждая сторона $П_2$ используется для разрезания исходного $П_1$, а не многоугольников, возникших в результате предыдущих разрезов. В конце процедуры точки пересечения, расположенные на сторонах $П_2$, группируются таким образом, чтобы было построено пересечение многоугольников. Исчерпывающий анализ этого метода выходит за пределы задач нашей книги, причем не только из-за ее сложности, но и потому, что в большинстве прикладных задач машинной графики многоугольник $П_2$ либо со-

ответствует наблюдаемому фрагменту изображения (некоторый правильный прямоугольник), либо является проекцией треугольника или четырехугольника (см. гл. 17).

15.6. ЭФФЕКТИВНОЕ ОПРЕДЕЛЕНИЕ ПЕРЕСЕЧЕНИЯ МНОГОУГОЛЬНИКОВ

Определение пересечения двух многоугольников путем многократного применения алгоритма 15.3, описанное в предыдущем разделе, не всегда оказывается эффективным. Так, информация относительно расположения вершин многоугольника Π_1, полученная за один прогон алгоритма, никак не используется при другом прогоне. Существуют различные способы увеличения эффективности этого процесса и некоторые из них будут здесь изложены. Напомним, что n и m обозначают число вершин (или сторон) многоугольников Π_2 и Π_1 соответственно. В наихудшем случае шаги *3*—8 алгоритма 15.3 выполняются mn раз. Минимальная оценка трудоемкости определения пересечения составляет $m + n$, поскольку все вершины должны быть обработаны, по меньшей мере, один раз. Разница между mn и $m + n$ может оказаться значительной, поэтому необходим поиск эффективных процедур.

Однако следует отметить, что, если m — малое число (скажем, 3 или 4), как это часто бывает в прикладных задачах машинной графики, повышение эффективности процедуры может оказаться невозможным. Экономия, полученная за счет уменьшения числа осмотров одной вершины, может утрачиваться вследствие роста объема учетных операций. Поэтому проводимое в этом разделе обсуждение, относится к задаче определения пересечения двух многочленов лишь в случае, когда у обоих число вершин значительно. Существенное значение для машинной графики имеют алгоритмы, обеспечивающие эффективное определение попарных пересечений для большого числа многоугольников. Именно такая ситуация возникает при решении задачи разделения видимых и невидимых частей воспроизводимого объекта, а также при проверке рисунка схемы, особенно в случае высокой степени интеграции. Здесь также применимы принципы, положенные в основу эффективных алгоритмов определения пересечения многоугольников. Поэтому мы опишем несколько основных применяемых методов.

Один из возможных способов ускорения определения пересечения связан с использованием фигур более простой формы, чем заданные многоугольники. Пусть M_1 и M_2 — правильные описанные прямоугольники для многоугольников Π_1 и Π_2 соответственно. (Их легко построить, используя экстремальные значения координат вершин многоугольников.) Вместо того, чтобы решать задачу пересечения для многоугольников, можно решать значительно более простую задачу определения пересечения прямоугольников M_1 и M_2. Если описанные прямоугольники не пересекаются, то это означает, что заданные многоугольники тем более не будут

пересекаться. Если один из прямоугольников находится внутри другого, например M_2 лежит внутри M_1, то можно рассматривать пересечение прямоугольника M_2 и многоугольника $П_1$ а затем пересечение полученного многоугольника и многоугольника $П_2$ (рис. 15.5,*а*). Если прямоугольники M_1 и M_2 имеют непустое пересечение M_{12}, то можно рассматривать пересечение каждого многоугольника с прямоугольником M_{12}, а получив два новых многоугольника, применять эту процедуру уже к ним (рис. 15.5,*б*). Преимуществом этого подхода является быстрое исключение из рассмотрения вершин многоугольников, далеко отстоящих друг от друга.

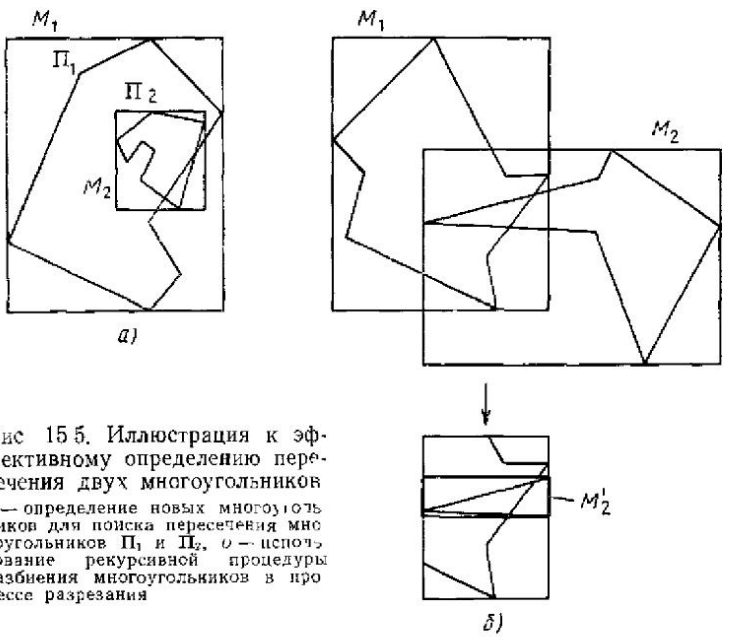


Рис 15 5. Иллюстрация к эффективному определению пересечения двух многоугольников

а — определение новых многоугольников для поиска пересечения многоугольников $П_1$ и $П_2$, *б* — использование рекурсивной процедуры разбиения многоугольников в процессе разрезания

Другие подходы предусматривают предварительную сортировку вершин, и наиболее экономичным оказывается случай, когда по крайней мере один многочлен остается фиксированным.

Первый метод, предложенный в статье [15.2], является весьма общим и подходит для решения различных геометрических задач. Рассмотрим множество прямых на плоскости $L_1, L_2, \ldots, L_k$ и точки их попарного пересечения $P_1, P_2, \ldots, P_h$. Очевидно, что $h<k(k + + 1)/2$. Обозначим координаты точки P_i через x_i, y_i. Без всякой потери общности можно допустить, что точки пересечения снабжены индексами таким образом, что $x_i<x_{i+1}$. Вертикальные прямые $x=x_i$ делят плоскость $h+1$ полос таким образом, что в пределах каждой полосы отсутствуют пересечения прямых. Теперь отрезки прямых можно упорядочить в пределах каждой полосы в соответствии, например, со значением координат y их средних точек. Это упорядочение является хорошо определенным в связи с

отсутствием пересечений отрезков в пределах полосы. Затраты на упорядочение K объектов составляют величину порядка $K log_2 K$ и в данном случае они будут равны $h\ log_2 h + hk\ log_2 k$.

Предположим далее, что задана новая точка (x, y) и требуется определить, находится ли она на одной из k прямых или лежит внутри области, задаваемой этими прямыми. Чтобы решить эту задачу, можно сначала найти полосу, в которой расположена эта точка, используя значение ее координаты x, а затем определить ее положение относительно прямых, исходя из ее координаты y. *Оба* решения эквивалентны введению объекта в упорядоченный список, т. е. задаче, для решения которой требуется время, пропорциональное длине списка[1]. В нашем случае это время равно $log_2 h + log_2 k$. Поскольку $log_2 h <= 2 log_2 k$ можно сказать, что затраты пропорциональны $log_2 k$. (Переборный алгоритм требует затрат порядка k.) В результате может быть сформулировано формальное утверждение.

Теорема 15.1. Определить положение точки относительно k прямых можно за время порядка $log_2 k$ при условии предварительной сортировки этих прямых. Затраты на предварительную сортировку определяются величиной порядка $k^2 log_2 k$ [15.2].

Другой метод, предложенный в докладе [15.3], предусматривает выполнение аналогичной предварительной сортировки, но на этот раз в полярных координатах. На использование метода наложено жесткое ограничение, требующее выпуклости обоих многоугольников. Метод заключается в выборе внутри одного из многоугольников, например $П_2$, точки C и определении полуотрезков $L_1, L_2, .., L_m$, соединяющих точку С с m вершинами многоугольника $П_2$. Затраты на отыскание точки и прямых пропорциональны m. Если прямые направлены из центра в вершину, то точкам, расположенным справа, соответствует положительное значение в уравнении (15.1) и, следовательно, легко установить, принадлежит ли точка сектору. Точка принадлежит сектору, образованному прямыми L_i и L_{i+1}, если соответствующее значение положительно относительно первой прямой и отрицательно относительно второй (рис. 15.6,*а*). Секторы можно упорядочить, например, по часовой стрелке, и, следовательно, можно найти сектор, которому принадлежит точка, затратив на это время, пропорциональное $log_2 m$. После нахождения сектора, содержащего одну вершину второго многоугольника, например $П_1$, для остальных вершин секторы можно найти, последовательно просматривая секторы и выясняя, не содержат ли они каких-либо вершин. Если многоугольник $П_1$ содержит центр (рис. 15.6,*б*), его секторы и вершины обходятся одновременно и никакие возвраты не предусматриваются. Если же центр лежит вне $П_1$ (рис. 15.6,в), то можно найти новый сектор, включающий $П_1$ и имеющий центр в

[1] Предполагается, что список реализуется с помощью адекватной структуры данных, например такой, как некоторое сбалансированное дерево. Более подробные сведения можно найти в монографии [6 1, гл. 3]

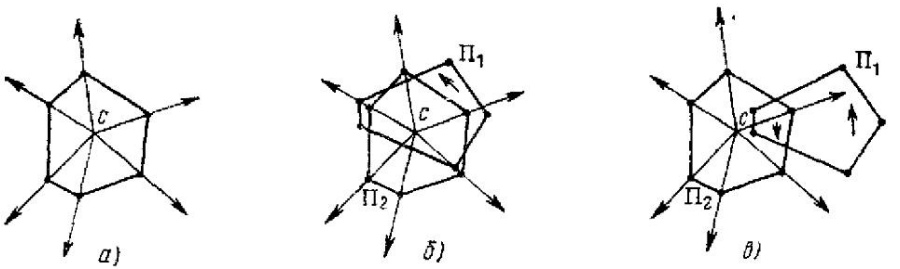


Рис. 15.6. Иллюстрация к эффективному определению пересечения двух многоугольников:

а — определение секторов в рамках процедуры определения пересечения двух выпуклых многоугольников, *б* — определение пересечения в случае, когда центр *C* расположен внутри обоих многоугольников, *для* обоих многоугольников направление последовательного обхода вершин может быть одним и тем же, *в* — определение пересечения в случае, когда центр *C* находится вне многоугольника $П_1$. Обход вершин многоугольника $П_1$ должен проводиться последовательно в двух направлениях, однако осмотру подлежит лишь часть секторов многоугольника $П_2$

точке *C*. После этого вершины многоугольника $П_1$ разделяются на две группы, каждая из которых при поиске соответствующих секторов может обрабатываться последовательно. В любом случае на этот процесс затрачивается время, пропорциональное *n*, где я — число вершин многоугольника $П_1$ После окончания этой процедуры оценивается положение вершин, расположенных в каждом секторе, относительно стороны многоугольника $П_2$, находящейся в этом секторе.

15.7. БИБЛИОГРАФИЧЕСКИЕ ЗАМЕЧАНИЯ

В монографии [1.4] описан алгоритм разрезания произвольного правильного прямоугольника с помощью произвольной прямой, в котором используется рекурсивное разбиение многоугольника на меньшие многоугольники В статье [15.4] проведено подробное рассмотрение алгоритма 15.3 и некоторых связанных с ним проблем. В статье [15.5] обсуждается общая задача преобразования многоугольников: разделение многоугольника на части, объединение двух пересекающихся многоугольников и т. д. Такие задачи, естественно, можно решать без особых затруднений, произведя соответствующие изменения в алгоритме разрезания, однако отсутствие правильного метода, позволяющего избежать многократного просмотра отдельных частей многоугольника, может привести к избыточным вычислениям.

Использование разбиения многоугольника на выпуклые компоненты при решении задач распознавания образов рассмотрено в монографии [37].

Теорема 15.1 представляет собой перефразировку теоремы, сформулированной Добкиным и Липтоном [15.2]. Второй из эффективных алгоритмов определения пересечения многоугольников предложен Шамосом [15.3]. Хотя этот и другие аналогичные ему

алгоритмы обладают асимптотически (при стремлении числа вершин многоугольника и бесконечности) более высоким быстродействием, чем алгоритмы, описанные в (нашей книге, возможность их непосредственного использования в машинной графике и распознавании образов вызывает определенные сомнения по причинам, приведенным в разд. 15.6 Кроме того, многоугольники со значительным числом вершин (скажем, больше 10), которые встречаются в задачах машинной графики, обычно не являются выпуклыми Таким образом, асимптотические эффективные алгоритмы для решения задачи пересечения многоугольников имеют значение для машинной графики лишь в случае, если они пригодны для работы с невыпуклыми многоугольниками С другой стороны, использование предварительной сортировки и описанных прямоугольников может оказаться целесообразным в задачах на пересечение очень большого числа многоугольников При решении таких задач можно воспользоваться простым методом типа алгоритма 15.3 для определения истинного пересечения, а асимптотически эффективный алгоритм следует использовать для уменьшения числа рассматриваемых многоугольников Методы, обнаруживающие, а не вычисляющие пересечения, изложены в докладе [15'1].

15.8. ЗАДАЧИ

15 1. Исследуйте алгоритм разрезания для частного случая, когда прямая L проходит через одну из вершин многоугольника Необходимо ли внести в алгоритм какие-либо изменения?

15.2. Какие изменения следует внести в алгоритм 15 1, если отрезок прямой заменяется произвольной (бесконечной) прямой, заданной ее коэффициентами?

15.3 Напишите программу, реализующую алгоритм 15.2

15.4 Измените алгоритм 153 таким образом, чтобы он обеспечивал прове дение вектора между последовательными точками пересечения многоугольника с прямой В таком случае фигура, полученная в результате определения пере сечения, будет замкнута и для ее заполнения можно воспользоваться одним *А3* алгоритмов, приведенных в гл. 8

Указание Необходимо использовать флаг, показывающий, какая сторона многоугольника в данный момент обрабатывается При изменении значения флага соответствующая точка пересечения заносится в память При очередной смене значения флага между запомненной и новой точками пересечения проводится вектор Следует обратить внимание на такие случаи, как расположение вершины многоугольника на прямой, а также на задание исходных данных для алгоритма Полезно выполнять обход многоугольника по часовой стрелке.

15.5 Напишите и реализуйте алгоритм для определения пересечения и объе динения двух многоугольников Результаты его работы должны представляться в виде множества многоугольников (В случае пересечения это множество мо жет быть пустым)

15.6. Напишите и реализуйте алгоритм для разбиения (невыпуклого) мно гоугольника на выпуклые подмножества Используйте советы, приведенные в разд. 15.5, и докажите, что объединение многоугольников, полученных в ре зультате разбиения исходного многоугольника, действительно представляет со бой исходный многоугольник Оцените вычислительную сложность Вашего алго ритма

157. Напишите и реализуйте алгоритм для разбиения (невыпуклого) мно гоугольника на выпуклые подмножества при наличии дополнительного ограни чения, запрещающего пересечение выпуклых компонент Сравните вычислитель-

ную сложность данного алгоритма и алгоритма, написанного в соответствии с задачей 15.6.

15.8 Определите точно, сколько раз приходится вычислять величины, определяемые уравнениями (15.1) или (152), при нахождении пересечения двух пятиугольников. Сделайте это сначала с помощью метода, изложенного в разд. 15.5, а затем с помощью одного из методов, изложенных в разд 15.6

Глава 16

МАТЕМАТИЧЕСКИЕ АСПЕКТЫ ТРЕХМЕРНОЙ МАШИННОЙ ГРАФИКИ

16.1. ВВЕДЕНИЕ

Машинная графика служит интересным примером возрождения некоторых почти преданных забвению методов и разделов математики. Проблема переноса трехмерных сцен на плоскую поверхность заботила живописцев на протяжении многих столетий, а изучение перспективы (центральной проекции) занимало видное положение в программах художественных учебных заведений. В начале девятнадцатого века изучение *начертательной геометрии* превратилось в основной предмет забот техники в связи с возникновением таких задач, как определение пересечения цилиндров средствами графики и т. п. Изобретение фотографии привело к падению практического значения реалистической живописи, а появление абстрактной живописи отодвинуло изучение проекций на второй план. Аналогичным образом, начертательная геометрия потеряла свое доминирующее положение в технике и к 1960 году в Соединенных Штатах лишь очень небольшое число студентов технических специальностей изучало этот предмет[1].

Машинная графика, задачей которой является получение двухмерных представлений для трехмерных сцен, оживила интерес к этим предметам. К важнейшим математическим средствам, необходимым для воспроизведения изображений, выглядящих трехмерными, относятся методы решения геометрических задач (см разд 16.2), позиционные проеобразования (см. разд. 163) и проекции (см. разд. 16.4 и 16.5)

Математика является необходимым, но не достаточным средством для получения хороших изображений. Другими необходимыми предпосылками являются наличие хорошей технической квалификации и определенной искушенности в области изобразительных искусств. В последние 10—15 лет наблюдается значительный

[1] Мне в определенном смысле повезло, что высшее образование я получил в стране, в которой программа обучения изменилась очень мало В пятидесятых годах в Афинском техническом университете начертательной геометрии уделялось серьезное внимание

интерес к явлению, известному как «машинная живопись». В основном, произведения машинной живописи — двухмерные. Вероятно, наибольший интерес с технической точки зрения представляет собой использование вычислительных машин для воспроизведения сцен, которые выглядят трехмерными. При решении таких задач требуется значительный объем вычислений, что, естественно, служит серьезным препятствием; усиление внимания к изучению математических средств, используемых при их решении, может привести к разработке более эффективных алгоритмов и программ.

16.2. ОДНОРОДНЫЕ КООРДИНАТЫ

Анализ, проведенный в разд. 14.3, можно обобщить на трехмерный случай. Точку P можно задать с помощью однородных координат x, y, z, w, а плоскость A — коэффициентами a, b, c, d, так

$$ax + by + cz + dw = 0. \tag{16.1}$$

В трехмерном случае двойственность имеет место между точкой и плоскостью. Таким образом, три точки задают плоскость посредством уравнения вида

$$\det \begin{vmatrix} x & x_1 & x_2 & x_3 \\ y & y_1 & y_2 & y_3 \\ z & z_1 & z_2 & z_3 \\ w & w_1 & w_2 & w_3 \end{vmatrix} = 0, \tag{16.2}$$

а три плоскости пересекаются в точке, координаты которой задаются минорами определителя:

$$\det \begin{vmatrix} a & a_1 & a_2 & a_3 \\ b & b_1 & b_2 & b_3 \\ c & c_1 & c_2 & c_3 \\ w & w_1 & w_2 & w_3 \end{vmatrix} = 0. \tag{16.3}$$

что уравнение плоскости принимает вид

Эти определения теряют силу в вырожденных случаях; три точки лежат на прямой, три плоскости имеют общую прямую, две точки или плоскости, задающие прямую, совпадают. В уравнении (16.2) условия вырожденности проявляются как возможность представить один из столбцов 2—4 в виде линейной комбинации двух других столбцов; это значит, что данный определитель равен нулю при любых значениях (x, y, z, w).

Итак, для задания прямых теперь требуется пара уравнений: прямая может определяться либо как пересечение двух плоскостей, либо как отрезок, соединяющий две точки. Чтобы записать уравнение прямой, соединяющей две точки, в однородных координатах, воспользуемся тем обстоятельством, что такая прямая должна принадлежать всем плоскостям, задаваемым точками (x_1, y_1, z_1, w_1), (x_2, y_2, z_2, w_2) и произвольной третьей точкой. Следователь-

но, уравнение (16.2) должно являться тождеством относительно значений элементов четвертого столбца. Это, в свою очередь, означает, что миноры, соответствующие элементам этого столбца, должны быть равны нулю. Однако не все четыре результирующие уравнения являются независимыми. В самом деле, рассматривая минор, соответствующий x_3, обнаруживаем, что строка (w, w_1, w_2) должна быть линейно зависима от строк (y, y_1, y_2) и (z, z_1, z_2), чтобы определитель был равен нулю. Аналогичным образом устанавливается линейная зависимость строки (w, w_1, w_2) от строк (x, x_1, x_2) и (z, z_1, z_2). Из этих двух обнаруженных линейных зависимостей следует линейная зависимость строк (x, x_1, x_2), (y, y_1, y_2) и (z, z_1, z_2). Следовательно, минор, соответствующий w_3, равен нулю, если равны нулю миноры, соответствующие x_3 и y_3. Это доказательство можно повторить для минора, соответствующего z_3.

$$\det \begin{vmatrix} x & x_1 & x_2 \\ y & y_1 & y_2 \\ w & w_1 & w_2 \end{vmatrix} = \det \begin{vmatrix} x & x_1 & x_2 \\ z & z_1 & z_2 \\ w & w_1 & w_2 \end{vmatrix} = 0. \qquad (16.4)$$

Если в этих уравнениях задать $w = w_1 = w_2 = 1$, то придем к обычным выражениям, используемым для задания отрезка прямой, соединяющего две точки:

$$(x - x_1)/(x_2 - x_1) = (y - y_1)/(y_2 - y_1) = (z - z_1)/(z_2 - z_1). \qquad (16.5)$$

так что в результате остаются два независимых уравнения:

16.2.1. ПОЛОЖЕНИЕ ПРОИЗВОЛЬНОЙ ТОЧКИ ОТНОСИТЕЛЬНО ПРОИЗВОЛЬНОЙ ПЛОСКОСТИ

Для того чтобы оценить положение некоторой точки относительно некоторой плоскости, необходимо определить знак определителя уравнения (16.1) для координат соответствующей точки. Эта задача в трехмерном случае оказывается более сложной, чем в двухмерном, и поэтому необходимо сделать ряд предварительных замечаний.

Определение 16.1. Пусть v_1, v_2 и v_3 — векторы, имеющие общую начальную точку и не принадлежащие одной и той же плоскости. Будем считать, что эти векторы образуют *правостороннюю систему,* если они обладают следующим свойством. Пусть P — плоскость, задаваемая векторами v_1 и v_2, и H — полупространство, задаваемое плоскостью P и содержащее вектор v_3. В таком случае для наблюдателя, находящегося в полупространстве H в точке, общей для всех трех векторов, совмещение векторов v_1 и v_2 будет выглядеть как поворот против часовой стрелки.

К получению такой системы приводит выбор обычной системы координат *x, y, z.* Общепринятым примером правой системы координат служат большой, указательный и средний пальцы, отождествляемые с векторами v_1, v_2 и v_3 соответственно. Когда пальцы устанавливаются так, что оказываются взаимно перпендику-

лярными, они образуют правую систему. Следующая лемма указывает простой аналитический способ проверки правосторонности системы.

Лемма 16.1. Значение определителя

$$d = \det \begin{vmatrix} x_1 & x_2 & x_3 \\ y_1 & y_2 & y_3 \\ z_1 & z_2 & z_3 \end{vmatrix} \tag{16.6}$$

положительно, если три соответствующие точки образуют *правостороннюю систему* относительно начала координат.

Доказательство. Без всякой потери общности ось x можно расположить вдоль прямой, соединяющей начало координат с первой точкой, ось y — на плоскости, задаваемой двумя первыми точками и образующей угол, меньший 90°, с прямой, соединяющей начало координат со второй точкой, и ось z — по нормали к этой плоскости, причем таким образом, чтобы она вместе с двумя другими осями образовывала правую систему координат (рис. 16.1). В таком случае уравнение (16.6) принимает вид

$$d = \det \begin{vmatrix} x_1 & x_2 & x_3 \\ 0 & y_2 & y_3 \\ 0 & 0 & z_3 \end{vmatrix} = x_1\, y_2\, z_3. \tag{16.6'}$$

Благодаря выбранному нами направлению осей координат значения всех трех координат, образующих правую систему, положительны и, следовательно, положительно их произведение, стоящее в правой части уравнения (16.6').

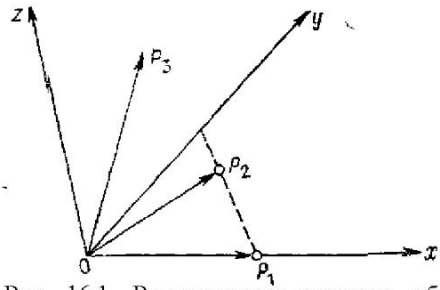


Рис. 16.1. Расположение векторов, образующих правостороннюю систему

Можно показать, что значение определителя d равно умноженному на шесть объему пирамиды, основанием которой служит треугольник, образованный тремя рассматриваемы ми точками, а вершиной — начало координат [16.1, p. 21].

Введем аналог допущения 14.1.

Допущение 16.1. Если плоскость задана тремя своими точками, го эти точки упорядочены таким образом, что они образуют правостороннюю систему относительно начала координат.

Утверждение 16.1. Пусть X, Y, Z и W — координаты точки P и (x_i, y_i, z_i, w_i), i=1, 2, 3 — три точки, задающие плоскость. Если W и все W_i имеют положительные значения, то точка P лежит с той же стороны данной плоскости, что и начало координат при условии положительности определителя:

$$D = \begin{vmatrix} X & x_1 & x_2 & x_3 \\ Y & y_1 & y_2 & y_3 \\ Z & z_1 & z_2 & z_3 \\ W & w_1 & w_2 & w_3 \end{vmatrix}. \qquad (16.7)$$

Доказательство. Поскольку правая часть уравнения (16.7) представляет собой линейную функцию от x, y, z и w, знак этой функции постоянен по каждую сторону плоскости, определяемой уравнением $D = 0$. Подстановка в уравнение (16.7) заачений $X=Y = Z = 0$ дает

$$D = W d, \qquad (16.8)$$

где d — определитель (16.6). В силу допущения 16.1, леммы 16.1 *Y* допущения о знаке *W оба* сомножителя в произведении (16.8) положительны.

Последний результат допускает обобщение, позволяющее использовать его для определения расположения произвольной точки относительно неплоских поверхностей. В самом деле, почти любую поверхность можно разбить на треугольники, вершины которых можно разметить некоторым единообразным способом так, чтобы они образовывали правостороннюю систему относительно начала координат. Единственным исключением являются поверхности типа *листа Мёбиуса,* которые в прикладных задачах машинной графики встречаются нечасто (более подробные сведения об ориентации поверхности можно найти в монографии [16.2, p. 214]).

16.2.2. ПЕРЕСЕЧЕНИЕ ТРЕУГОЛЬНИКОВ

Введем следующее обозначение определителя:

$$S_{ijkn} = \begin{vmatrix} x_i & x_j & x_k & x_n \\ y_i & y_j & y_k & y_n \\ z_i & z_j & z_k & z_n \\ w_i & w_j & w_k & w_n \end{vmatrix}. \qquad (16.9)$$

Утверждение 16.2. Рассмотрим два треугольника, каждый из которых задается тремя точками: первый треугольник — точками P_1, P_2, P_3, второй —точками P_4, P_5, P_6. Пусть x_i, y_i, z_i и w_i — однородные координаты точки P_i ($1<=i<=6$). В таком случае два треугольника пересекаются, если не все значения определителей S_{1456}, S_{2456} и S_{3456} имеют один и тот же знак и не все значения определителей S_{4123}, S_{5123} и S_{6123} имеют один и тот же знак.

Доказательство. Это утверждение является непосредственным следствием утверждения 16.1.

Данный результат представляет собой обобщение анализа, проведенного в подразд. 14.4.2 для двухмерного случая, на трехмерный случай.

16.3. ТРЕХМЕРНЫЕ ПРЕОБРАЗОВАНИЯ

Преобразования, обеспечивающие перенос и изменение масштаба, в трехмерном случае можно представлять таким же образом, как и в двухмерном, однако для поворота теперь следует указывать ось, относительно которой он осуществляется. Хотя формулы для преобразования координат в результате поворота относительно произвольной оси довольно громоздки, им можно придать аккуратный вид с помощью достаточно простых операций, если воспользоваться некоторыми понятиями векторной алгебры

16.3.1. НЕКОТОРЫЕ СВЕДЕНИЯ ИЗ ВЕКТОРНОЙ АЛГЕБРЫ

На протяжении всей книги предполагалось, что читатели знакомы с элементами векторной алгебры; мы уже пользовались понятиями скалярного произведения и произведения матрицы на векторы (см. разд. 2.2, разд. 10.3 и др.). В этом разделе будет введен еще ряд операций над векторами, которые широко используются в механике и электромагнитной теории, однако могут быть не очень хорошо знакомы студентам, специализирующимся в области информатики или обработки сигналов.

Определение 16.2. Векторное произведение трехмерных векторов и и v (обозначается через uXv) определяется как вектор ,с компонентами

$$u_y v_z - u_z v_y,\ u_z v_x - u_x v_z,\ u_x v_z - u_z v_x. \qquad (16.10)$$

Следующее утверждение суммирует ряд свойств векторного произведения. Соответствующие доказательства очевидны.

Утверждение 16.3. а. Если векторы и и v коллинеарны, т. е. u = *c*v, где *c* — константа, то uXv = 0

б Если векторы и и v неколлинеарны, то вектор uXv нормален к плоскости, определяемой этими векторами.

в uXv = —vXu.

г. Значение векторного произведения равно |u|*|v|sinθ, где θ — угол, образованный перемножаемыми векторами

Определение 16.3. Векторным произведением вектора v л матрицы *A* является матрица, столбцы которой представляют собой векторные произведения столбцов матрицы *A* на вектор v.

Отметим также, что произведение *n*-мерного вектора v на транспонированный *n*-мерный вектор и представляет собой матрицу *W*, элементы которой определяются уравнением

$$W_{ij} = v_i u_j. \qquad (16.11)$$

Поскольку вектор-столбец представляет собой матрицу размерами *nX1*, а вектор-строка — матрицу 1Xn, то указанное векторное произведение представляет собой матрицу *nXn* (При изменении порядка сомножителей результатом векторного произведения является матрица 1X1, т. е. скалярная величина.) Эта операция над векторами специально выделяется.

Определение 16.4. *Диадой* называется произведение вектора-столбца и на вектор-строку v': uv'. *Двучленным* представлением матрицы A является форма

$$A = \mathbf{a}_1 \mathbf{a}_1' + \mathbf{a}_2 \mathbf{a}_2' + \ldots + \mathbf{a}_k \mathbf{a}_k', \qquad (16\ 12)$$

где a_1 a_2,..., a_k — векторы.

Понятие диады нами будет использовано при выводе выражения для преобразования координат в результате поворота относительно оси, проходящей через начало координат. Обсуждение двучленного представления выходит за пределы задач нашей книги, и мы ограничимся рассмотрением одного примера. Обратите внимание на то, что не все матрицы имеют двучленное представление.

Пример 16.1. Матрица

$$A = \begin{bmatrix} 2 & 0 \\ 1 & 3 \end{bmatrix}$$

имеет двучленное представление

$$A = \begin{bmatrix} 1 \\ 0 \end{bmatrix} (2 - 2) + \begin{bmatrix} 3 \\ 3 \end{bmatrix} (0\ 1).$$

С другой стороны, матрицу

$$A = \begin{bmatrix} 2 & 1 \\ 0 & 2 \end{bmatrix}$$

нельзя представить в двучленном виде.

16.3.2. ПОВОРОТ ОТНОСИТЕЛЬНО ПРОИЗВОЛЬНОЙ ОСИ, ПРОХОДЯЩЕЙ ЧЕРЕЗ НАЧАЛО КООРДИНАТ

Рассмотрим эту задачу сначала в координатах x, y, z. Зададим ось, проходящую через начало координат, углами φ_1, φ_2 и φ_3, которые она образует с координатными осями x, y и z. Отметим, что эти углы не являются независимыми, поскольку их косинусы связаны следующим соотношением:

$$\cos^2 \varphi_1 + \cos^2 \varphi_2 + \cos^2 \varphi_3 = 1.$$

Определим вектор a, параллельный заданной оси и имеющий единичную длину. При этом ось $a_i = \cos \varphi_i$, $i = 1, 2, 3$.

Теорема 16.1. Матрица преобразований координат, соответствующих повороту на угол 9 относительно оси, проходящей через начало координат и параллельной единичному вектору a, определяется уравнением

$$R(\theta) = \mathbf{a}\mathbf{a}' + \cos\theta\,(I - \mathbf{a}\mathbf{a}') + \sin\theta\,(I \times \mathbf{a}), \qquad (16.13)$$

где I — единичная матрица 3x3.

Доказательство 16.1. Пусть b — вектор, проходящий через начало координат и подлежащий повороту относительно заданной оси. Этот вектор b всегда можно разложить на два вектора, один

из которых параллелен заданной оси, а второй (d) нормален к ней; таким образом, вектор b можно представить в виде суммы

$$\mathbf{b} = \lambda\,\mathbf{a} + \mathbf{d}, \tag{16.14}$$

где λ — постоянная. Определим произведение $R\mathbf{b}$, воспользовавшись уравнениями (16.13) и (16.14):

$$R\,\mathbf{b} = \lambda\,\mathbf{a}\mathbf{a}'\mathbf{a} + \lambda\cos\theta\,(\mathbf{a} - \mathbf{a}\mathbf{a}'\mathbf{a}) + \lambda\sin\theta\,(I\times\mathbf{a})\,\mathbf{a} + \mathbf{a}\mathbf{a}'\mathbf{d} + \\ + \cos\theta\,(\mathbf{d} - \mathbf{a}\mathbf{a}'\mathbf{d}) + \sin\theta\,(I\times\mathbf{a})\,\mathbf{d}. \tag{16.15}$$

Все операции над векторами ассоциативны и произведение диады ef' на вектор g равно произведению вектора e на скалярное произведение f'd. Применив такое преобразование ко всем членам уравнения (16.15), можно добиться существенного упрощения последнего. Поскольку, по предположению, вектор a — единичный, произведение a'a=1, и поскольку вектор d был задан ортогонально вектору a, произведение a'd = 0. Кроме того, произведение *(Ixa)*a должно быть равно нулю, поскольку все столбцы матрицы IXa ортогональны вектору a. В таком случае уравнение (16.15) принимает вид

$$R\,\mathbf{b} = \lambda\,\mathbf{a} + \cos\theta\,\mathbf{d} + \sin\theta\,(I\times\mathbf{a})\,\mathbf{d}. \tag{16.16}$$

Для завершения доказательства теоремы необходимо лишь показать, что два последних члена этого уравнения представляют поворот вектора d на угол 0 в плоскости, перпендикулярной заданной оси Во-первых, обратим внимание на то, что (IXa)d = dXa. Это можно

$$I\times\mathbf{a} = \begin{bmatrix} 0 & a_3 & -a_2 \\ -a_3 & 0 & a_1 \\ a_2 & -a_1 & 0 \end{bmatrix}$$

и вычислив затем значение ее произведения с вектором d. Итак, два последних члена уравнения (16.16) можно представить в виде

$$\cos\theta\,\mathbf{d} + \sin\theta\,\mathbf{d}\times\mathbf{a}. \tag{16.17}$$

доказать, определив в явном виде матрицу

Вектор dXa перпендикулярен векторам d и a и его длина равна длине вектора d в силу утверждения 163г. Если на плоскости, перпендикулярной заданной оси, задать локальную систему координат, то одну из осей можно выбрать параллельной вектору d, а другую — параллельной вектору dXa. В результате оказывается, что уравнение (16.17) действительно представляет поворот в в этой плоскости на угол 6. На этом доказательство теоремы заканчивается.

Уравнение (16 16) можно преобразовать таким образом, чтобы матрица поворота оказалась выражена через углы 9 и φ_i . В нее можно ввести также дополнительные строку и столбец, чтобы она была применима при использовании однородных координат. В таком случае поворот на угол 0 будет задаваться матрицей 4X4, элементы которой определяются уравнениями (16.18). В этих урав-

нениях величина k принимает значения из множества $\{1, 2, 3\}$, отличные от значений i и j:

а) при $1 \leqslant i, j \leqslant 3$

$$R_{ij} = \cos\varphi_i \cos\varphi_j (1 - \cos\varphi) - \cos\varphi_k \sin\theta (-1)^{i+j}, \text{ если } i < j; \quad (16.18а)$$

$$R_{ij} = \cos\varphi_i \cos\varphi_j (1 - \cos\varphi) + \cos\varphi_k \sin\theta (-1)^{i+j}, \text{ если } i > j; \quad (16.18б)$$

$$R_{ii} = \cos^2\varphi_i (1 - \cos\theta) + \cos\theta; \quad (16.18в)$$

б) при $i=4$ или $j=4$

$$R_{i4} = 0 \quad 1 \leqslant i \leqslant 3, \quad (16.18г)$$

$$R_{4j} = 0 \quad 1 \leqslant j \leqslant 3, \quad (16.18д)$$

$$R_{44} = 1. \quad (16.18е)$$

Если выбранная ось вращения совпадает с одной из осей координат, то приведенные выше уравнения существенно упрощаются и сводятся, в сущности, к формуле, определяющей поворот на плоскости.

16.4. ОРТОГОНАЛЬНЫЕ ПРОЕКЦИИ

Проекции, возможно, представляют самый важный класс преобразований, используемый в машинной графике, поскольку они обеспечивают получение двухмерных изображений трехмерных объектов Наиболее простой разновидностью проекций является *ортогональная проекция,* при использовании которой изображение некоторой точки определяется как точка пересечения перпендикуляра, опущенного из этой точки на плоскость проекции, с плоскостью проекции. Если последняя представляет собой плоскость λ—y, то величина z просто приравнивается нулю (или к какой-либо иной постоянной).

Пример 16.2. Рассмотрим куб, заданный восемью вершинами A (0,0,0), B(2,0,0), C*(2,2,0),* D(0,2,0), E(0,0,2), F(2,0,2), G(2,2,2) и H(0,2,2) Требуется определить ортогональную проекцию куба после его поворота на 60° относительно оси, соединяющей вершины (0,0,0) и (2,2,2).

В данном случае cos $=\varphi_1 =$ cos $\varphi_2 =$ cos $\varphi_3 = 1/\mathrm{sqrt}(3)$, и первая группа элементов матрицы поворота определяется для любого угла θ следующими уравнениями:

$$R_{ij} = \frac{1}{3}(1 - \cos\theta) - \frac{1}{\sqrt{3}} \sin\theta (-1)^{i+j}, \text{ если } i < j;$$

$$R_{ij} = \frac{1}{3}(1 - \cos\theta) + \frac{1}{\sqrt{3}} \sin\theta (-1)^{i+j}, \text{ если } i > j;$$

$$R_{ii} = \frac{1}{3}(1 - \cos\theta) + \cos\theta.$$

Для $\theta = 60°$ матрица поворота имеет следующий вид:

$$R = \begin{bmatrix} 2/3 & 2/3 & -1/3 \\ -1/3 & 2/3 & 2/3 \\ 2/3 & -1/3 & 2/3 \end{bmatrix} \quad (16.19)$$

Здесь не выписаны последние столбец и строка матрицы R, поскольку они должны определяться из уравнений (16 18г—е), а в данном примере однородные координаты не используются. Вершины рассматриваемого куба теперь имеют координаты.

$$A\,(0,\ 0,\ 0),\ B\left(\frac{4}{3},\ -\frac{2}{3},\ \frac{4}{3}\right),\ C\left(\frac{8}{3},\ \frac{2}{3},\ \frac{2}{3}\right),$$

$$D\left(\frac{4}{3},\ \frac{4}{3},\ -\frac{2}{3}\right),$$

$$E\left(-\frac{2}{3},\ \frac{4}{3},\ \frac{4}{3}\right),\ F\left(\frac{2}{3},\ \frac{2}{3},\ \frac{8}{3}\right),$$

$$G\,(2,\ 2,\ 2)\ \text{и}\ H\left(\frac{2}{3},\ \frac{8}{3},\ \frac{2}{3}\right).$$

Проекция куба на плоскость 2=0 будет определяться следующими точками:

$$A\,(0,\ 0),\ B\left(\frac{4}{3},\ -\frac{2}{3}\right),\ C\left(\frac{8}{3},\ \frac{2}{3}\right),\ D\left(\frac{4}{3},\ \frac{4}{3}\right),$$

$$E\left(-\frac{2}{3},\ \frac{4}{3}\right),\ F\left(\frac{2}{3},\ \frac{2}{3}\right),\ G\,(2,\ 2)\ \text{и}\ H\left(\frac{2}{3},\ \frac{8}{3}\right).$$

Эти Точки приведены на рис. 16.2. Для правильного построения проекций необходимо установить, какая из точек — D *(4/3, 4/3)* и F *(2/3, 2/3)* — является видимой. Первая из них имеет координату z= - 2/3, а вторая — z=8/3.

При взгляде на куб сверху видна точка с большим значением z — этому случаю соответствует рис. 16.3,а; противоположный случай представлен на рис. 16.3,*б*.

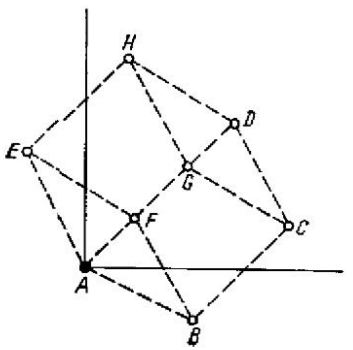


Рис 16.2. Расположение вершин куба, рассматриваемого в примере 16.2

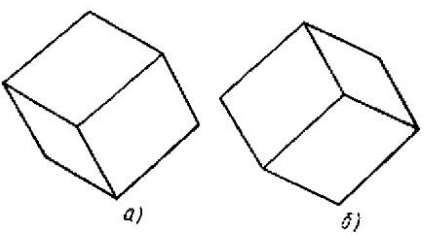


Рис. 16.3. Проекции куба, рассматриваемого в примере 16.2:
а — вид сверху, *б* — вид снизу

Хотя строить ортогональные проекции нетрудно, они не позволяют получать реальные сцены, если только не вводится допущение об очень значительном удалении наблюдателя от объектов наблюдения В случаях, когда расстояние, разделяющее наблюдателя и сцену, сопоставимо с размерами представленных на ней объектов, следует использовать центральные проекции.

16.5. ЦЕНТРАЛЬНЫЕ ПРОЕКЦИИ

Введение центральной проекции представляет собой попытку имитировать способ, с помощью которого наблюдатель формирует представление о наблюдаемой им сцене. Объекты проектируются на картинную плоскость из центральной точки — глаза наблюдателя (рис. 16.4).

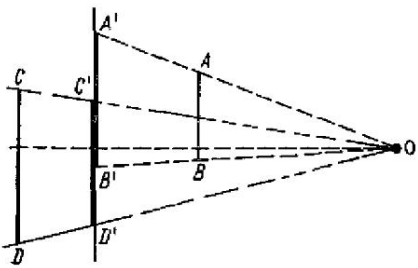


Рис. 16.4. Иллюстрация к понятию центральной проекции. Проекцией отрезка AB служит отрезок $A'B'$, а отрезка CD — отрезок $C'D'$

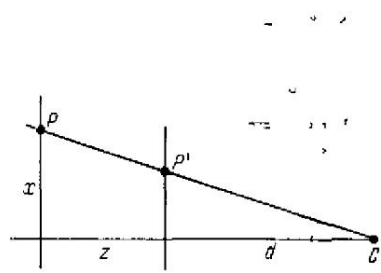


Рис 16.5. Соотношение координат проектируемой точки и ее центральной проекции

Подобное спроектированное изображение легко построить, опираясь на описание объемного тела; этот способ изображения используется в живописи на протяжении многих веков. Следует отметить, что лишь очень небольшая часть художников слепо придерживается законов перспективы, поскольку изображения, нарисованные в соответствии с ними, не всегда выглядят правильно. В машинной графике, к сожалению, необходимо строго следовать определенному правилу, поскольку программы вычислительных машин в состоянии реализовать лишь математические предписания. Если мы не удовлетворены результатами применения строгих законов перспективы, следует внести в них изменения, сформулированные математически в явном виде. Центр проектирования обычно совмещается с позицией наблюдателя, который смотрит на экран, расположенный на расстоянии d от центра проектирования; последний можно использовать и в качестве начала координат. Применив обозначения, приведенные на рис. 16.5, устанавливаем, что точка P с координатами x, y, z отображается в точку P' с координатами $xd/(d+z)$, $yd/(d+z)$, 0. Однако при воспроизведении изображения необходимо учесть загораживание одних объектов другими и, следовательно, необходимо сохранять информацию о значениях координаты z. По причинам, которые будут изложены ниже, для координаты г масштаб изменяется таким же образом, как и для координат x и y. *Эта* операция не повлияет на относительные расстояния объектов от экрана, так как при $z_1>z_2>0$ справедливо также неравенство $dz_1/(d+z_1) > dz_2/(d+z_2) > 0$.

Перспективные преобразования допускают компактные представления при использовании однородных координат, поскольку в

этом случае приходится изменять лишь последнюю компоненту. Можно, в частности, записать, что

$$P = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 1/d & 1 \end{bmatrix}. \qquad (16\ 20)$$

Если (ξ, η, ζ, w) — однородные координаты, то после проектирования

$$w_p = w + \zeta/d = w\,(1 + z/d) = w\,(d + z)/d,$$

так как $\zeta = wz$. Отсюда определяется искомое значение масштабного множителя. Для воспроизведения на экране некоторой оценки, необходимо сначала умножить все ее точки на матрицу P, затем определить видимые точки, после чего высветить те точки экрана, координаты x и y которых соответствуют координатам видимых объектов.

Пример 16.3. Воспроизводится изображение, имитирующее вид, открывающийся перед водителем, едущим по прямой дороге. Будем считать, что водитель находится в начале координат и направление движения перпендикулярно экрану. Поскольку система координат также движется в этом направлении, различные изображения можно получать, меняя значения координаты z объектов сцены. Для простоты будем считать, что на сцене, кроме дороги и горизонта, имеется всего один объект — квадратный рекламный щит, расположение которого показано на рис. 16.6,*а* и *б*. Этот щит образует с осью дороги угол 45°, его левый край находится на расстоянии u от прямой, по которой ведет машину водитель, высота щита равна и, ширина — $sqrt(2v)$ и расстояние между

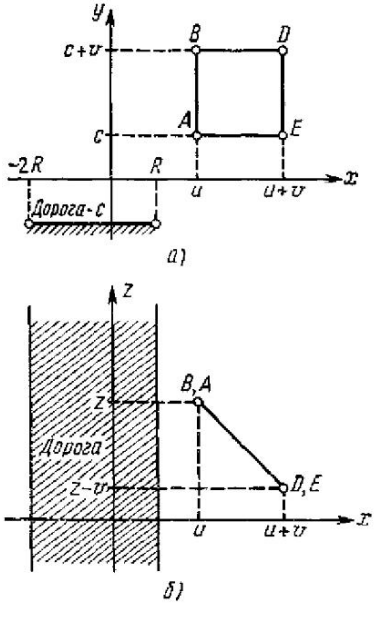


Рис. 16.6. Расположение объектов, рассматриваемых в примере 16.3:
а — в плоскости x-y, *б* — в плоскости x-z, *в* — полученная проекция

его нижним углом и водителем равно c. В таком случае углы щита имеют координаты:

$$A=\begin{bmatrix} u \\ c \\ z \end{bmatrix} \quad B=\begin{bmatrix} u \\ c+v \\ z \end{bmatrix} \quad D=\begin{bmatrix} u+v \\ c+v \\ z-v \end{bmatrix} \quad E=\begin{bmatrix} u+v \\ c \\ z-v \end{bmatrix}. \qquad (16\,21)$$

Расстояние между поверхностью дороги и водителем равно c, ее правым краем и водителем — R, и левым краем дороги и водителем — $2R$. Таким образом, две параллельные прямые, обозначающие дорогу, определяются следующей парой уравнений

$$x=+R, \quad y=-c \text{ (правый край);} \qquad (16\,22а)$$

$$x=-2R, \quad y=-c \text{ (левый край).} \qquad (16\,22б)$$

Отметим, что отдельные части изображения не будут изменяться. Точка, расположенная на правой границе дороги на расстоянии z от водителя, воспроизводится как точка с координатами.

$$x=Rd/(d+z), \quad y=-cd/(d+z). \qquad (16\,23)$$

:Отсюда следует, что при z-> • , $x=y=0$ и при г=0 $x=R$ и У= - c. Таким образом, правый край дороги будет все время воспроизводиться как отрезок прямой, соединяющий точки с координатами $(R, —c)$ и (0,0). Аналогичным образом, левый край дороги будет воспроизводиться как отрезок прямой, соединяющий точки $(—2R, —c)$ и (0,0). Соответствующее изображение представлено на рис. 16.6,*в*. Координаты центральных проекций углов рекламного щита определяются следующими выражениями

$$A_p=\begin{bmatrix} ud/(d+z) \\ cd/(d+z) \end{bmatrix}, \quad B_p=\begin{bmatrix} ud/(d+z) \\ (c+v)\,d/(d+z) \end{bmatrix}, \qquad (16\,24а)$$

$$D_p=\begin{bmatrix} (u+v)\,d/(d+z-v) \\ (c+v)\,d/(d+z-v) \end{bmatrix}, \quad E_p=\begin{bmatrix} (u+v)d/(d+z-v) \\ cd/(d+z-v) \end{bmatrix}. \qquad (16\,24б)$$

При z->• все точки стягиваются к началу координат. Если изображение воспроизводится с разрешением, равным N пикселам, то наименьшим различимым значением является $1/N$ и, следовательно, при $c<u$ наибольшим значением z, дающим при проекции ненулевые координаты, является $d(Nc—1)$. Программа, обеспечивающая воспроизведение такого изображения, содержит цикл, начинающий работу с $z=d(Nc—1)$ и осуществляющий вычисления для последовательно уменьшающихся значений z вплоть до 0. Следует уделить внимание отсечению части изображения, так как точки E и D первыми покидают пределы экрана. Если можно организовать доступ к словам памяти обновления изображения (см. разд. 1.6 и 17), то выбранный участок изображения можно стереть с экрана и затем воспроизвести его снова, не затрагивая фон. Изображение рекламного щита можно получить, воспользовавшись каким-либо простейшим алгоритмом заполнения области и применив его ко всему изображению.

Этот пример показывает, что построение центральных проекций не вызывает затруднений в случаях, когда отсутствует взаимодействие изображаемых объектов. Если T — матрица межкадровых перемещений и P — проекционная матрица, то процесс проектирования складывается из умножения всех (представленных в однородных координатах) векторов и на произведение матриц $P*T$, разрезания сцены в соответствии с формой окна наблюдения и, в конце концов, воспроизведения полученных результатов. Задача усложняется, если объекты сцены взаимодействуют друг с другом, при этом приходится решать задачу удаления невидимых линий.

16.6. БИБЛИОГРАФИЧЕСКИЕ ЗАМЕЧАНИЯ

Вывод формул для поворота относительно произвольной оси основывается на материале учебника, надо признать, довольно старого [16.4]. Очевидно из-за сложности эти формулы отсутствуют в большинстве современных учебников — даже в книгах, посвященных машинной графике, они, насколько можно судить, игнорируются (например, в монографии [1.4]). Даже когда они приводятся, их вывод осуществляется путем трудоемких преобразований, но не с помощью элегантного двучленного представления, примененного Уилсоном и Гиббсом [16.4]. Проективная геометрия, что и не удивительно, располагает многочисленными средствами изучения проекций. Читатель должен иметь в виду, что в литературе по проективной геометрии связь некоторого объекта с его центральной проекцией называют просто перспективой [14.1, 14.4—14.7] [1]. Результаты, получаемые при построении центральных проекций, часто противоречат привычным представлениям людей (см. задачу 16.8), поскольку они искажают метрические соотношения. В монографии [16.1] приведен ряд очень остроумных примеров использования метода перспективы, приводящих к получению «невозможных» сцен.

16.7. ЗАДАЧИ

16.1. Выведите уравнения (16.2) и (16.3).

16.2. (а) Покажите, что площадь треугольника, вершины которого имеют координаты $x_1, y_1, z, x_2 y_2, z$ и $x_3 y_3 z$, определяется следующим выражением:

$$\text{площадь} = 0{,}5\,[x_1 (y_2 - y_3) + x_2 (y_3 - y_1) + x_3 (y_1 - y_2)].$$

(б) Выведите выражения для площади треугольника в случае, когда не **все** вершины имеют одинаковые значения координаты г.

16.3. Дайте подробное доказательство утверждения 16.2.

16.4. Дайте доказательство для всех четырех частей утверждения !6.3.

16.5. Определите матрицу поворота для осуществления поворота на неко торый угол относительно оси *у*.

16.6. Определите преобразование, обеспечивающее построение ортогональ ной проекции на плоскость, заданную тремя точками: (1,0,0), (0,1,0) и (0,0,1).

Указание: при решении этой задачи может оказаться полезной матрица (16.19).

16.7. Покажите, что каждая из четырех точек A_p, B_p, D_p и E_p, рассмат ривавшихся в примере 163, перемещается по прямой, проходящей через точку (0,0), при изменении значения z.

Покажите, что ортогональная проекция сферы всегда представляет собой круг, а ее центральной проекцией может быть либо круг, либо эллипс. Последнее для многих людей неочевидно. Это один из примеров отличия ме тода перспективы, используемого в живописи, от математического понятия перспективы. Следующий эксперимент может помочь Вам убедиться в спра ведливости приведенного утверждения. Направьте в темной комнате электри ческий фонарь на баскетбольный мяч с близкого расстояния. Тень на стене комнаты будет иметь форму эллипса во всех случаях, за исключением того, что в литературе описано очень много алгоритмов разделения видимых и невидимых элементов, мы приведем здесь лишь когда прямая, соединяющая источник света и центр сферы (мяча), перпендикулярна стене.

16.8. 16.9. Предусмотрите в программе-редакторе точек, рассматривавшейся в гл. 10, средства для работы с трехмерными объектами. Для этого в программу необходимо ввести блоки для реализации трехмерных преобразований

[1] В отечественной литературе по математике в качестве основного термина используется «центральная (она же коническая) проекция». Иногда употребляется ее синоним «перспективная проекция», а иногда перспективу называют центральной проекцией и наоборот. — *Прим. перев.*

Глава 17

СИНТЕЗ ТРЕХМЕРНЫХ ИЗОБРАЖЕНИЙ В МАШИННОЙ ГРАФИКЕ

17.1. ВВЕДЕНИЕ

Употребляя термин *трехмерные изображения,* мы имеем в виду изображения, представляющие собой проекции трехмерных сцен, но не изображения, построенные в трех измерениях[1]. Тем не менее использование соответствующих операций проектирования, удаление поверхностей, которые, как считается, не могут быть видны, и применение штриховки позволяет создавать впечатление подлинной глубины, как в этом можно убедиться, обратившись к рис. 17.1 и 17.2. При синтезе изображений такого рода в центре внимания оказывается разделение видимых и невидимых элементов сцены, а также алгоритмы штриховки (см. разд. 13.9) и методы рандомизации, позволяющие придать воспроизводимому изображению текстурированный вид, как показано на рис. 17.2. Эффективное решение проблемы разделения видимых и невидимых элементов, вероятно, является одним из решающих этапов при синтезе трехмерных изображений. Проблемы видности объектов изображения часто решаются как задачи удаления *невидимых линий* и задачи удаления *невидимых поверхностей.* Общему рассмотрению этой темы посвящен разд. 17.2. В разд. 17.3 обсуждается алгоритм разделения видимых и невидимых элементов объекта, основанный на использовании тетрадного дерева, а в разд. 17.4 приводится алгоритм, которым удобно пользоваться при работе на растровых графических устройствах. В разд. 17.5 рассматривается проблема *однородности* по наблюдаемости: каким образом можно ускорить разделение видимых и невидимых элементов объекта, учитывая взаимозависимость отдельных частей сцены. Несмотря на то, что в литературе описано очень много алгоритмов разделения видимых и невидимых элементов, мы приведем здесь лишь

[1] Истинно трехмерные изображения можно строить, опираясь на голографические средства (обсуждение голографии выходит за пределы задач нашей книги). Стереографические изображения требуют использования двух проекций, каждая из которых может быть получена методами, описанными в данной главе, хотя мы и не останавливаемся на подробностях, касающихся точного согласования этих проекций. Соответствующий пример можно найти в докладе [17.7].

два простейших. Дело в том, что выбор конкретного метода разделения в очень существенной степени зависит от характера прикладной задачи. Проблема здесь заключается главным образом в объединении некоторых из рассмотренных выше методов, в частности алгоритмов разрезания многоугольников и алгоритмов за-



Рис. 17.1. Сцена, имеющая естественный вид, воспроизведена с помощью штриховки и отражения по описанию, включающему ряд многоугольников и 36 бикубических участков поверхностей, ограниченных замкнутыми кривыми (Воспроизводится из доклада S M Rubin and T. Whitted A 3-Dimensional Representation for Fast Rendering of Complex Scenes. — *SIGGRAPH'80* p. 110—116)

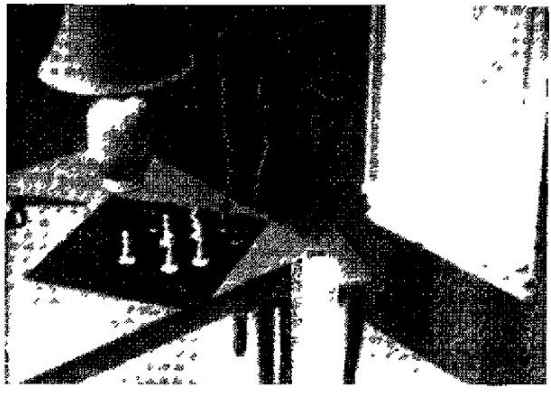


Рис. 17.2. Сцена, которую трудно отличить от написанной картины, получена посредством использования штриховки, удаления невидимых поверхностей и центральной проекции. Описания объектов сцены включают 32 тыс многоугольников.
(Воспроизводится из доклада T. Whitted and D M Weimer A Software Test-Bed for the Development of 3-D Raster Graphics Systems — *SIGGRAPH'81,* p. 271 —

полнения области. Изменениям, которые необходимо внести в эти алгоритмы для того, чтобы их можно было применять к описаниям нелинейных объектов, посвящен разд. 17.6. И, наконец, в разд. 17.7 кратко рассматриваются рандомизация и многократное отражение света.

В принципе, разделение видимых и невидимых элементов сцены можно осуществлять как в пространстве объектов, так и в пространстве изображений. В первом случае определение того, что видно и что не видно, производится на основе геометрических соотношений между частями объектов. Во втором случае осуществляется обход изображения и изучение проекций по мере их построения. Оба алгоритма, описанные в разд. 17.3 и 17.4, работают в пространстве изображений. В разд. 17.5 рассмотрен пример решения проблемы видности в пространстве объектов.

При воспроизведении изображений сложных объектов необходимо располагать соответствующими математическими описаниями этих объектов. Для их получения обычно применяются два приема. Первый предусматривает вычерчивание множества кривых на реальном трехмерном объекте (рис. 17.3,*а*) и использова-

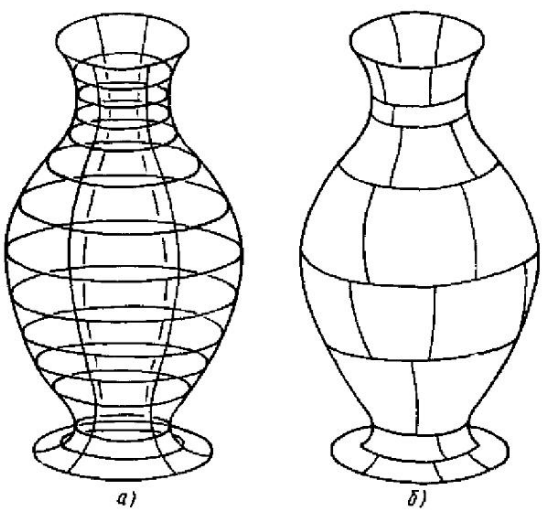


Рис. 17.3. Иллюстрация способов, используемых для получения математического описания сложных трехмерных объектов

а — представление тела множеством кривых и б — множеством конечных участков поверхности, ограниченных замкнутыми кривыми

ние математических описаний этих кривых для воспроизведения изображения. Этим методом удобно пользоваться при работе на векторных графических устройствах, однако изображения, которые воспроизводятся при его применении, напоминают проволочные каркасы.

Второй прием предусматривает разбиение поверхности тела на некоторое число участков, ограниченных замкнутыми кривыми. Для каждого из таких участков можно найти простое приближенное математическое описание, и изображение объекта воспроизводится как совокупность изображений этих конечных участков поверхности, ограниченных замкнутыми кривыми. На рис. 17.3,*б* приведен соответствующий пример. В качестве таких конечных участков часто используются плоские треугольники, поскольку триангуляция поверхности — довольно простая процедура, ибо задание трех точек позволяет определить поверхность. Кроме того, воспроизведение изображений треугольников, их заполнение и определение попарных пересечений не вызывает ни малейших затруднений. Основной недостаток такого способа представления трехмерных объектов заключается в том, что для создания впечатления гладкости поверхности может потребоваться очень много треугольников. Для описания объекта можно использовать и конечные участки поверхности тех типов, которые были рассмотрены в разд. 13.4—13.7, а именно на них ориентируются самые современные методы воспроизведения изображений трехмерных сложных объектов.

17.2. ЗАДАЧИ УДАЛЕНИЯ НЕВИДИМЫХ ЛИНИЙ И НЕВИДИМЫХ ПОВЕРХНОСТЕЙ

Изображение куба, полученное в примере 16.2, иллюстрирует простейший случай задачи удаления невидимых линий. Хотя общее число прямых в этом случае равно 12, для создания впечатления трехмерного объекта необходимо воспроизвести лишь девять из них. Эту задачу можно также решать посредством удаления невидимых поверхностей, если рассматривать грани куба. В векторной графике обычно решают задачу удаления невидимых линий, а в растровой графике — задачу удаления невидимых поверхностей. Важность этой задачи определяется не присущей ей сложностью, а тем, что ее решение необходимо получить быстро, на пример, в процессе поворота объекта. Первый шаг решения этой задачи связан с обобщением результатов, приведенных в подразд. 14.4.4, на трехмерный случай с помощью результатов из разд. 16.2.

17.2.1. ТЕНИ ПОВЕРХНОСТЕЙ

Пусть $S_1, S_2, \ldots, S_n$ — грани (конечные участки поверхности) всех объектов, присутствующих на изображении.

Определение 17.1. Мы будем считать, что грань S_i, затеняет или загораживает грань S_j, по направлению v при выполнении следующих условий:

а) грань S_i не пересекает грань S_j;

б) ортогональные проекции этих граней T_i и T_j на некоторую плоскость, перпендикулярную направлению v, имеют некоторое непустое пересечение T_{ij};

в) по меньшей мере, одна из точек грани S_i, проекция которой входит в T_{ij}, находится дальше от начала координат по направлению v, чем соответствующая точка грани S_j.

Обычно предполагается, что направление v совпадает с осью z, а грани представляют собой плоские многоугольники. Если не вводить последнее допущение, то проверка выполнения условий определения 17.1 оказывается очень трудной, если вообще практически невозможной. Сравнение этого определения с утверждением 14.2 показывает, насколько усложняется эта задача в трехмерном случае: вместо проверки выполнения условий (14.21*а*) или (14.21*б*) приходится решать задачу разрезания многоугольников на плоскости x—y. Проверка отсутствия пересечения тоже достаточно сложна Отрезок прямой на плоскости однозначно определяется заданием двух его концевых точек, и мы можем непосредственно воспользоваться анализом, проведенным в подразд. 14.4.2. В трехмерном пространстве грань не обязательно является треугольной, поэтому недостаточно проверить знак определителя (16.9) для трех точек — его необходимо проверять для всех вершин обеих граней. Положение точки относительно начала координат можно определить, подставив координаты одной из точек из плоскостей в уравнение, задающее другую, на которой введена правая система координат.

Изменим определение 17.1 таким образом, чтобы оно учитывало случай центральной проекции.

Определение 17.2. Мы будем считать, что грань S_i затеняет или загораживает грань 5, для наблюдателя, расположенного в точке C, при выполнении следующих условий:

а) грань S_i не пересекает грань S_j;

б) центральные проекции этих граней T_i и T_j на некоторую пло скость, не проходящую через точку С, имеют некоторое непустое пересечение T_{ij},;

в) по меньшей мере, одна из точек грани S_i, проекция которой входит в T_{ij} лежит на прямой, проходящей через точку C, даль ше от этой точки, чем соответствующая точка грани S_j.

Различия, связанные с типом проекции, можно устранить, если воспользоваться выражениями для центральных проекций, приведенными в разд. 16.5. В этом случае сцена, воспроизводимая с помощью центральных проекций, строится в два этапа. Сначала все объекты подвергаются трехмерному преобразованию, задаваемому матрицей P [см. (16.20)]. В результате точка с координатами x, y, z отображается в точку с координатами

$$x' = dx/(z+d),\ y' = dy/(z+d),\ z' = dz/(z+d). \qquad (17.1)$$

На рис. 17.4,*а* приведена геометрическая интерпретация этих преобразований на плоскости x—z. Из элементарной геометрии известно, что

$$A'K/AL = CK/CL = d/(d+z),$$

так как $AL = x$ и $A'K$—x'. Кроме того,

$$A'^*A'' / A^*'N = CK/CL = d / (d + z),$$

так как $A'N=KL = z$ и $A'A'' = z'$. Таким образом, точка A'' является образом точки Л при данном преобразовании. На рис. 17.4,*б* показано преобразование группы точек. Из уравнения (17.1), как

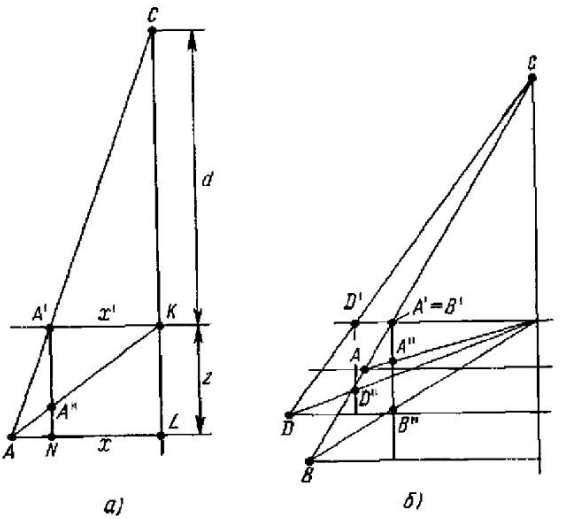


Рис. 17.4. Иллюстрация к задаче разделения видимых и невидимых элементов сцены при центральном проектировании:
а — геометрическая интерпретация преобразования, задаваемого матрицей (16.20), точка A' — образ точки A при этом преобразовании, б — геометрическая интерпретация того же, что и на рис 17.4,а, преобразования для группы точек

и из рис. 17.4, следует, что образы точек, расположенных на одной прямой, проведенной из центра проектирования *С* (таких, как точки Л и *В),* выстраиваются вдоль вертикали картинной плоскости (такие, как точки *А"* и *В").* Образы точек, расположенных на разных прямых, проведенных из центра проектирования С (таких, как точки *А* и *D),* не выстраиваются вертикально (такие, как точки *А"* и *D").*

После этого можно переходить ко второму этапу. В процессе его выполнения рассматриваются тени поверхностей, но делается это не для исходных объектов, а для их образов при преобразовании, задаваемом матрицей (16.20), и при ортогональном проектировании Соответствующие результаты суммируются следующим образом.

Утверждение 17.1. Решение задачи разделения видимых и невидимых элементов сцены при центральном проектировании эквивалентно решению этой задачи для образов исходных объектов при преобразовании, задаваемом матрицей (16.20), и при ортогональном проектировании.

17.2.2. ПОДХОДЫ К РЕШЕНИЮ ЗАДАЧИ РАЗДЕЛЕНИЯ ВИДИМЫХ И НЕВИДИМЫХ ЭЛЕМЕНТОВ СЦЕНЫ

Определения 17.1 и 17.2 предусматривают проверку всех пар граней при выявлении тех, которые загораживают другие грани. Это нереально при работе с изображениями, содержащими тысячи подобных граней, но осуществимо, если речь идет о гранях отдельного объекта. Действительно, пусть сцена содержит k объектов и число поверхностей i-го объекта равно S_i. В таком случае сложность полного упорядочения пропорциональна величине

$$\left[\sum_{i=1}^{k} s_i\right]^2, \tag{17.2}$$

а сложность процедуры сравнения поверхностей лишь одного объекта пропорциональна величине

$$\sum_{i=1}^{k} s_i^2. \tag{17.3}$$

Поскольку обычно объекты не пересекаются, можно решать задачу затенения поверхностей для их очертаний; сложность решения при этом пропорциональна величине k^2. Этот подход требует тщательного ведения учетных операций. Во многих алгоритмах разделения видимых и невидимых элементов сцены это затруднение преодолевается посредством превращения трехмерной задачи в две двухмерные, для чего используется один из следующих способов:

а) Рассматриваются многоугольники, являющиеся проекциями граней, и с помощью какого-либо алгоритма разрезания выявляются все пересекающиеся пары многоугольников (в том числе такие пары многоугольников, в которых один полностью покрывается другим). Затем с помощью утверждения 16.1 устанавливается, какой из многоугольников пары загораживает другой многоугольник. Вместо проверки непосредственно пар можно рассматривать их пересечение с окнами, задаваемыми правильными многоугольниками. Два многоугольника могут загораживать друг друга лишь в том случае, если они пересекаются с одним и тем же окном. Этот метод положен в основу алгоритма 17.1.

б) Рассматривается пересечение всех граней с некоторой горизонтальной плоскостью и для полученных в результате отрезков прямых решается задача разделения видимых и невидимых элементов. Этот метод реализован в алгоритме 17.2.

17.2.3. НАБЛЮДАЕМОСТЬ НА ИЗОБРАЖЕНИИ ОТДЕЛЬНОГО ВЫПУКЛОГО ОБЪЕКТА

Если на изображении имеется лишь один объект, причем выпуклый, то задачу разделения видимых и невидимых элементов можно решать, определяя направление нормалей к граням объекта. Эти направления можно находить с помощью методов, изложенных в разд. 13.2. Если нормаль обращена в сторону, про-

тивоположную экрану, то соответствующая грань расположена на задней стороне объекта и потому не видна. Если же нормаль обращена к экрану, то соответствующая грань объекта видна. (Направление нормали легко установить по знаку величины F_z, определяемой уравнением (13.5в).) В разд. 17.5 обсуждается феномен геометрической однородности (однородности по наблюдаемости) применительно к ряду аналогичных методов.

17.3. АЛГОРИТМ РАЗДЕЛЕНИЯ ВИДИМЫХ И НЕВИДИМЫХ ЭЛЕМЕНТОВ СЦЕНЫ, ОСНОВАННЫЙ НА ИСПОЛЬЗОВАНИИ ТЕТРАРНОГО ДЕРЕВА

Этот алгоритм обрабатывает фрагменты изображения, соответствующие вершинам тетрарного дерева (см. разд. 6.4). Если в пределах некоторого фрагмента границы объекта отсутствуют, то цвет этого фрагмента можно определить с помощью простого упорядочения по направлению z. Именно так обстоит дело с фрагментами 11, 22, 24, 42, 44 и 33 на рис. 17.5 (эти фрагменты относятся

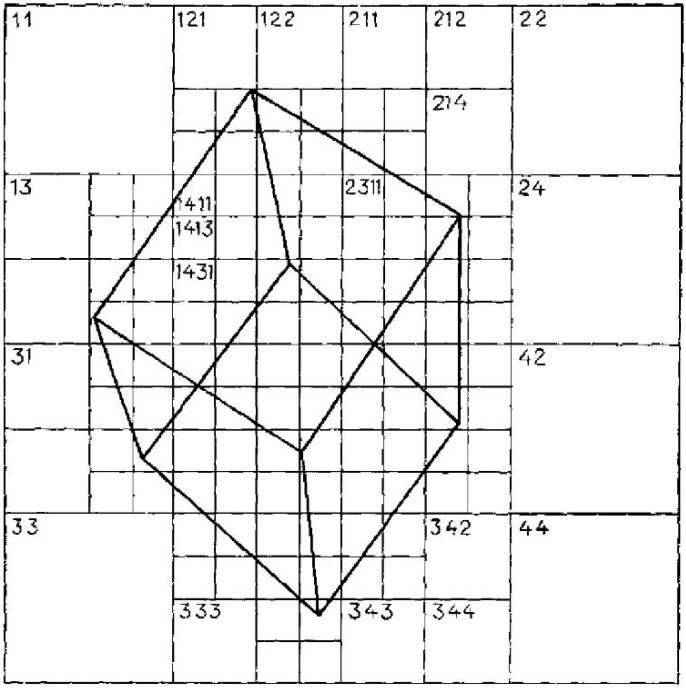


Рис. 17.5. Иллюстрация к работе алгоритма удаления невидимых линий, основанного на использовании тетрарного дерева. Жирными линиями изображены проекции границ объекта, а тонкими линиями выделены квадратные наблюдаемые фрагменты изображения, соответствующие первым трем уровням тетрарного дерева

к фону изображения). Если же на фрагменте обнаруживается граница, то он подвергается разбиению на подфрагменты посредством спуска по тетрарному дереву. На рис. 17.5 представлено разбиение, соответствующее спуску на два уровня относительно исходного разбиения изображения на фрагменты. В число однородных в результате этого разбиения попадают фрагменты 1413, 1431 и т. д. В конечном счете последовательное разбиение приводит к уровню, соответствующему отдельным пикселам. Пикселам, расположенным на границе, значение (яркость или цвет) может присваиваться в зависимости от значений соседних пикселов. Этот процесс, естественно, очень напоминает процедуру сегментации, в которой в качестве предиката однородности области используется высказывание «отсутствие границы в области». (Исторически дело обстоит как раз наоборот, см. разд. 17.8). Границы, которые не видны, устраняются автоматически, поскольку области, расположенные по обе стороны от них, имеют один и тот же цвет, а именно — цвет загораживающей поверхности.

Основная часть этого алгоритма связана с определением того, пересекается ли грань с наблюдаемым фрагментом изображения (окном). Существуют три возможности: а) грань покрывает фрагмент изображения; б) грань пересекается с фрагментом изображения или расположена внутри него; в) грань и наблюдаемый фрагмент находятся в разных частях изображения и не пересекаются (рис. 17.6). Решение можно найти, решив задачу разрезания для границ F и фрагмента W с помощью применения алгоритма 15.2 ко всем сторонам грани. Эту процедуру можно упростить, так как алгоритму не нужна информация о результате пересечения многоугольников — важно знать пересекаются они или нет (см. задачу 17.3).

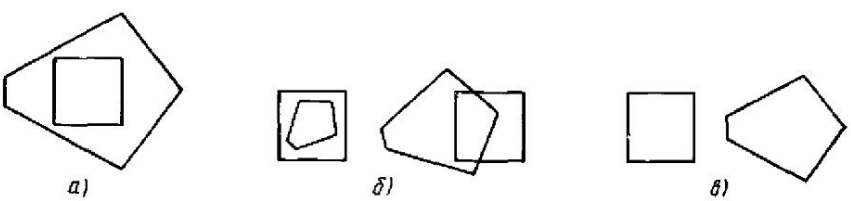


Рис. 17.6. Иллюстрация к работе алгоритма удаления невидимых линий, основанного на использовании тетрарного дерева: допустимые случаи взаимного расположения грани и наблюдаемого фрагмента изображения; значения величины K, получаемые в результате применения процедуры LOC, равны двум (а), единице *(б)* и нулю (в)

Рассмотрим процедуру $LOC(W, F)$, реализующую этот метод и выдающую в качестве результата следующие значения: 2, если имеет место случай а) и грань F покрывает фрагмент W, как показано на рис. 17.6,а; 1, если имеет место любая из двух конфигураций на рис. 17.6, *б*, т. е. грань F покрывается фрагментом W или пересекается с ним; 0, если имеет место случай в), т. е. грань

F и фрагмент W находятся на некотором расстоянии друг от друга (рис. 17.6,*в*).

Работа алгоритма начинается с того, что изображение разбивается на фрагменты, соответствующие листьям тетрарного дерева. Для каждого наблюдаемого фрагмента заводятся списки: список L содержит перечень всех граней, имеющихся на изображении и могущих иметь пересечение с соответствующей вершиной (фрагментом); список M содержит те грани, которые покрывают соответствующую вершину (фрагмент). В исходном состоянии список M — пустой, а список L содержит все грани; по окончании выполнения алгоритма пустым оказывается список L, а список M включает те грани, которые покрывают соответствующую вершину (фрагмент). При обработке каждого фрагмента алгоритм обращается к процедуре *LOC* при изучении каждой грани, включенной в список L (шаг 4 алгоритма 17.1). Вначале число таких граней очень велико, однако большая их часть удаляется, так как они расположены вне рассматриваемого фрагмента. Это удаление осуществляется на шаге 5. Если в результате использования процедуры *LOC* для обработки некоторого заданного фрагмента для всех граней выдается 0, то это означает, что цвет данного фрагмента совпадает с цветом фона и больше он не будет рассматриваться (флаг F оставляется равным 0). Если в результате применения этой процедуры выдается 2, то соответствующая грань переносится из списка L в список M (шаг 7). Если при обработке некоторого фрагмента результатами являются только 0 и 2, то этот фрагмент также можно исключить из дальнейшего рассмотрения. Он полностью покрывается одной или несколькими гранями и его цвет совпадает, следовательно, с цветом ближайшей по отношению к наблюдателю грани. Это можно установить на основании просмотра содержания списка M на шаге 8. Если для какой-либо грани процедура дает 1, то соответствующий фрагмент необходимо подвергнуть разбиению (шаг 6). Очевидно, что всякая грань, отстоявшая на некоторое расстояние от исходного фрагмента, будет отстоять и от фрагментов, являющихся его непосредственными потомками на тетрарном дереве. Поскольку первое разбиение на более мелкие фрагменты происходит в случае, когда результатом выполнения процедуры *LOC* является 1, то понятно, что любые изменения содержания списков L *к* M связаны с отстоящими или покрытыми гранями. Следовательно, эти списки могут быть переданы непосредственным потомкам разбиваемого фрагмента на тетрарном дереве. Грань, для которой процедура выдала 1, должна остаться в списке L для всех непосредственных потомков фрагмента [1].

В разд. 17.5 обсуждаются несколько изменений, которые можно использовать на этапе задания начальных условий алгоритма.

[1] При использовании алгоритмов, предусматривающих копирование, а не совместное использование списков, следует проявлять осторожность, поскольку после копирования списки каждого из непосредственных потомков вершин на тетрарном дереве могут изменяться по-разному.

Кроме того, работу алгоритма можно ускорить, изменив шаг 4 таким образом, чтобы грань, пересекающаяся с фрагментом, сравнивалась с элементами списка *M*. Если оказывается, что она расположена за одним из них, то проверяемая грань больше не рассматривается.

Алгоритм 17.1. Алгоритм удаления невидимых линий, основанный на использовании тетрарного дерева

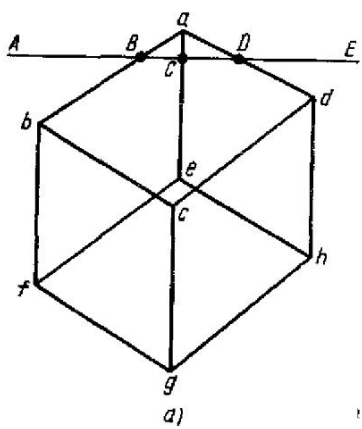
Обозначения. *Q* — квадратный фрагмент, соответствующий листу тетрарного дерева. Для каждого квадратного фрагмента алгоритм ведет два списка — *L* и *M*. *L* — список, содержащий все грани, которые могут иметь пересечение с фрагментом *Q*. *M* — список, содержащий грани, покрывающие фрагмент *Q* (это устанавливается в процессе выполнения алгоритма). Флагу *F* присваивается 1 для всех квадратных фрагментов, для которых задача разделения видимых и невидимых элементов еще не решена.

0. Выбирается начальный уровень тетрарного дерева и для каждого квадратного фрагмента, соответствующего вершине, заводится список *L*, в который помещаются все грани воспроизводимых трехмерных объектов. Список *M* для всех фрагментов — пустой. Массиву флагов *F* присваивается начальное значение 1.
1. For каждого квадратного фрагмента *Q*, для которого *F (Q)* равно 1, do:
   Begin.
2. *F (Q)* присваивается нулевое значение.
3. For каждой грани сцены *P* в списке *L(Q)* do:
   Begin.
4. Производится обращение к процедуре *LOC (Q, P)*.
   Пусть в результате ее выполнения *K*=0.
5. If значение *K* равно 0, then грань *P* удаляется из списка *L(Q)* {грань *P* расположена вне квадратного фрагмента Q}.
6. If значение *K* равно 1, then квадратный фрагмент Q заменяется своими непосредственными потомками на тетрарном дереве, которым передаются списки L и M фрагмента Q {грань *P* пересекается с квадратным фрагментом Q}. Флагу каждого из непосредственных потомков присваивается значение единица и break из цикла.
7. If значение *K* равно 2, then грань *P* заносится в список *M (Q)* и удаляется из списка *L(Q)*. {грань *P* покрывает квадратный фрагмент Q}.
   End.
   End.
8. For каждого квадратного фрагмента Q do:
   Begin.

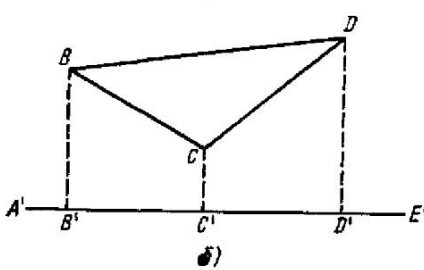
9. If список M — пустой, then воспроизводится цвет фона.
10. Else просматривается содержимое списка M с тем, что бы найти ближайшую к наблюдателю грань и воспроизвести ее цвет.
End.
11. Конец алгоритма.

17.4. РАСТРОВЫЙ АЛГОРИТМ ПОСТРОЧНОГО РАЗДЕЛЕНИЯ ВИДИМЫХ И НЕВИДИМЫХ ЭЛЕМЕНТОВ СЦЕНЫ

Растровый алгоритм строчной развертки решает задачу разделения видимых и невидимых элементов сцены в процессе построчного воспроизведения изображения. Если все объекты представляют собой плоские области и между ними отсутствуют пересечения, то данный алгоритм сводится к процедуре заполнения области между сторонами многоугольника, используемой в алгоритме 8.1 (см. разд. 8.2). В этом случае границы заполняемых контуров упорядочиваются по значениям координаты y[1]. После этого рассматриваются пересечения объектов с плоскостями, задаваемыми уравнениями y = постоянная величина. Для многогранников эти пересечения представляют собой многоугольники, расположенные в плоскости x-z, а отрезки прямых, для которых должно осуществляться заполнение, являются проекциями этих многоугольников на ось x.



а)



б)

Рис. 17.7. Иллюстрация к работе алгоритма построчного разделения видимых и невидимых элементов сцены:

а — проекция куба, изображение которого должно быть воспроизведено, и плоскости, задаваемой уравнением y = постоянная, на плоскость x-y, *б* — проекция фигуры, полученной в результате пересечения, на плоскость x-z и на строку развертки

На рис. 17.7,а изображен куб (спроектированный на плоскость x—y), ребра которого ab, ae и ad пересекаются плоскостью, задаваемой уравнением y = постоянная, в точках B, C и D соответственно. На рис. 17.7,*б* представлена проекция этого пересечения на плоскость x-z, а также проекции вершин треугольника на

[1] Предполагается, что координатная ось z перпендикулярна экрану, а ось y перпендикулярна направлению развертки по экрану.

плоскость экрана, которые расположены вдоль прямой $A'E'$. Этими проекциями являются точки B', C' и D'. В этом примере задача разделения видимых и невидимых элементов сводится к установлению того, будет ли алгоритм заполнения области заполнять сегмент $B'D'$ цветом грани *abcd* или, наоборот, будет заполнять сегменты $B'C'$ и $C'D'$ цветом граней *abfe* и *aehd* соответственно.

Этот подход реализован в алгоритме 17.2. Основную часть алгоритма составляют шаги 3—6; лаконичность описания алгоритма связана с тем, что на некоторых его шагах выполняются операции, которые были подробно описаны в предыдущей части нашей книги. Если воспроизводимая сцена изменяется во времени, то необходимо для каждого значения y найти пересечения всех граней с плоскостями, задаваемыми уравнениями y = постоянная. Эта операция выполняется на шаге 4. При ее реализации необходимо использовать алгоритм разрезания типа алгоритма 15.1 (или алгоритма 15.3, если грани невыпуклые). Разрезаемый многоугольник представляет собой проекцию грани на плоскость x-y, а разрезающая прямая определяется пересечением плоскости y = постоянная с плоскостью проекции. Результатом работы алгоритма является множество сегментов, расположенных на плоскости x-y.

С другой стороны, если одну и ту же сцену приходится воспроизводить многократно, то эффективность алгоритма можно повысить, представляя ему описания объектов как многоугольников, образуемых при пересечении соответствующих объектов с плоскостями, задаваемыми уравнениями y = постоянная. Более того, эти многоугольники можно упорядочить по значениям y. (Аналогично хранению контуров, упорядоченных по значениям y, как это предусмотрено в алгоритме 8.1.) В таком случае шаг 4 сводится к обычному просмотру списка, содержащего эти многоугольники.

После выполнения шага 4 необходимо установить, какие сегменты затеняют другие сегменты (эта процедура была рассмотрена в подразд. 14.4.4). Описанный там подход можно использовать непосредственно и получать в результате некоторый список видимых сегментов. В алгоритме 17.2 применен иной способ. Заводятся два массива *COLOR* (ЦВЕТ) и *DEPTH* (ГЛУБИНА), элементы которых соответствуют пикселам, расположенным вдоль строки развертки. В массиве *COLOR* хранится цвет определенного пиксела, а в массиве *DEPTH* — значение координаты z сегмента, расположенного ближе всех остальных по отношению к наблюдателю. Для каждой строки развертки в качестве начального значения массива *COLOR* задается цвет фона, а в качестве начального значения массива *DEPTH*—максимально допустимое значение координаты г. Алгоритм при обработке сегмента, расположенного на плоскости x—z (например, сегмента BC на рис. 17.7), сравнивает значения координаты z точек этого сегмента со значением x, соответствующим тому же значению координаты x и хранящимся в массиве *DEPTH*. Если значение z точки меньше

значения, хранящегося в массиве *DEPTH,* то последнее корректируется и цвет данной точки присваивается соответствующей точке массива *COLOR.* Очевидно, что по окончании обработки всех сегментов массив *COLOR* содержит точную копию сцены и может быть воспроизведен.

Алгоритм 17.2. Алгоритм удаления невидимой поверхности в процессе строчной развертки

Обозначения. Массив *COLOR* является буфером для хранения значения цвета воспроизводимого пиксела. Массив *DEPTH* содержит значение z точки, расположенной ближе всего к наблюдателю, для каждого значения x строки развертки. P — отрезок прямой, образованный пересечением грани со строкой развертки R. x_p и z_p — координаты точек p и P соответственно.

1. *For* каждой строки развертки R экрана **do:**
   Begin.
2. Массиву *COLOR* присваивается начальное значение цвета фона и массиву *DEPTH* — максимально допустимое значение 2.
3. For каждой грани F объекта сцены **do:**
   Begin.
4. If грань F пересекается с горизонтальной плоскостью, задаваемой уравнением $y = R$, **then** определяется их пересечение P и do:
   Begin.
5. For каждой точки p пересечения P do:
   Begin.
6. If значение z_p меньше, чем $D[x_p]$, **then** $C[x_p]$ присваивается цвет P и $D[x_p]$ присваивается значение z_p. **End.**
   End. End.
7. Воспроизводится содержимое массива *COLOR.*
   End.
8. **Конец алгоритма.**

17.5. ОДНОРОДНОСТЬ ЭЛЕМЕНТОВ ИЗОБРАЖЕНИЯ ПО НАБЛЮДАЕМОСТИ

Алгоритмы 17.1 и 17.2 неэффективны, поскольку они не используют никакую информацию относительно размера или взаимного расположения граней. Их можно применять для воспроизведения сцен, содержащих некоторое количество прямоугольных граней, расположенных случайным образом. Эта ситуация редко встречается в реальных прикладных задачах, в которых обычно между различными частями изображения устанавливаются многочисленные отношения. Такого рода взаимозависимость интерпретируют как *однородность элементов изображения по наблюдаемо-*

сти. Приведя различные типы однородности элементов изображения по наблюдаемости, используемые в машинной графике, мы кратко обсудим возможности улучшения соответствующих алгоритмов. Однородность по наблюдаемости можно определить целым рядом способов и задавать их можно как в пространстве объектов, так и в пространстве изображений. Укажем несколько возможных способов определения однородности по наблюдаемости.

Однородность границы по наблюдаемости. Наблюдаемость некоторой границы изменяется лишь при пересечении ее некоторой другой границей. Из этого свойства можно извлечь преимущества, сформировав список граничных сегментов, не имеющих пересечений. После этого приходится проверять наблюдаемость лишь одной точки каждого сегмента. Поскольку выявление пересечений связано с выполнением целого ряда операций, использование этого свойства дает преимущество лишь в случае многократного воспроизведения одной и той же сцены.

Однородность грани по наблюдаемости. В связи с тем, что размер граней многогранника мал по сравнению с изображением *в* целом, обычно оказывается, что если видна часть грани, то видна и вся грань.

Однородность объекта по наблюдаемости. Наблюдаемость некоторого объекта часто можно определить, рассматривая некоторое описывающее его тело, форма которого может быть весьма простой, — это может быть сфера или многогранник с гранями, параллельными координатным плоскостям. Это некоторое обобщение идеи, использованной для определения пересечения многоугольников в разд. 15.6, и применяться оно может целым рядом способов. Так, например, можно упростить шаг 4 алгоритма 17.2, заведя отдельный список для диапазонов изменения значений y (максимальное и минимальное значения) для каждого объекта. Можно, например, использовать однородность объекта по наблюдаемости в алгоритме 17.1 для уменьшения начального размера списка L. И, наконец, использование этого вида однородности по наблюдаемости может облегчить определение отношений затенения поверхностей, о чем шла речь в подразд. 17.2.2.

Однородность строк развертки по наблюдаемости. Сегменты, которые видны в одной строке, вероятно, видны и в следующей строке развертки. Если имеется оценка минимального размера объекта, то можно задать значение K и изменить алгоритм 17.2 следующим образом. После обработки и воспроизведения строки развертки l_i,- пропускаются K—1 строк и очередной обрабатываемой строкой служит l_{i+k}. Если одни и те же отрезки видны на обеих строках, обозначим через F множество граней, которым эти отрезки принадлежат. После этого для пропущенных строк формируется массив C, причем используются лишь грани, принадлежащие множеству F. Затем K строк развертки воспроизводятся одновременно. (Можно полностью обойти решение задачи разделения видимых и невидимых элементов для промежуточных строк при помощи интерполирования разбиений, заданных на строках

развертки l_i и $l_{i+к}$.) Если же отрезки, которые видны на строке li+к, отличаются от тех отрезков, которые видны на строке l_i, то можно осуществить поиск с возвратом к промежуточной строке, например строке $l_{i+к/2}$, и повторить эту процедуру заново. Соответствующий алгоритм может быть организован аналогично алгоритму построения многоугольников, рассмотренному в разд. 12.5. Этот подход позволяет получать правильные результаты, если ни один из объектов не оказывается столь малым, что его полностью покрывают K строк развертки.

Межкадровая однородность по наблюдаемости. Изображение не очень сильно изменяется при переходе от текущего кадра к следующему. Для эффективного использования этого вида однородности по наблюдаемости необходимо вести список перемещающихся и фоновых объектов. В этом случае алгоритм воспроизведения изображения должен при смене кадра обрабатывать лишь участок в окрестности движущихся объектов. Напомним также наше обсуждение обработки перемещающихся объектов, проведенное в разд. 1.6.

Геометрическая однородность по наблюдаемости. Число допустимых видимых и невидимых конфигураций для каждой вершины ограничено.

В машинной графике этому типу однородности по наблюдаемости уделяется мало внимания, однако применительно к распознаванию изображений (анализу сцен) эта тема оживленно обсуждается. Если известно, что тело выпуклое, то использование этого вида однородности по наблюдаемости приводит к получению очень простых правил разделения видимых и невидимых элементов. Если, в частности, максимальный угол, образованный ребрами, сходящимися в вершине, меньше 180° (рис. 17.8,*а*), то два крайние ребра видны, а наблюдаемость остальных ребер зависит от их взаимного расположения относительно плоскости, определяемой первыми двумя ребрами. Если же такой угол превышает 180°, то наблюдаемость всех ребер одинакова (рис. 17.8,*б*). Наилучший способ использования геометрической однородности по наблюдаемости заключается в том, что сначала задача разделения видимых и невидимых элементов решается для каждого объекта, в

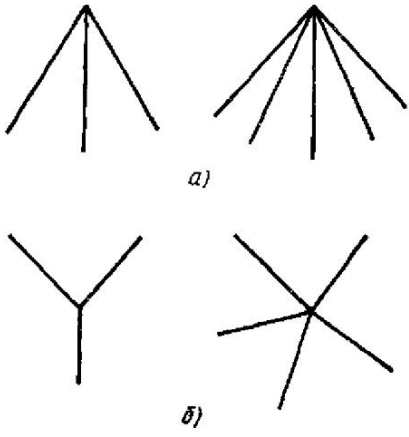


Рис. 17.8. Пример использования геометрической однородности по наблюдаемости:
а — если изображенные ребра принадлежат некоторому выпуклому объекту, то два крайние ребра должны быть видны; *б* — если изображенные ребра принадлежат некоторому выпуклому объекту, то все ребра, окружающие некоторую вершину, характеризуются одной и той же наблюдаемостью

результате чего ребра или грани этих объектов помечаются соответствующим образом. При воспроизведении изображения следует принимать во внимание лишь наблюдаемые грани каждого объекта и, следовательно, число элементов, подлежащих сортировке, уменьшается. Если вершины или ребра контура проекции (*ABCGHE* на рис. 17.9) снабжены специальными метками и

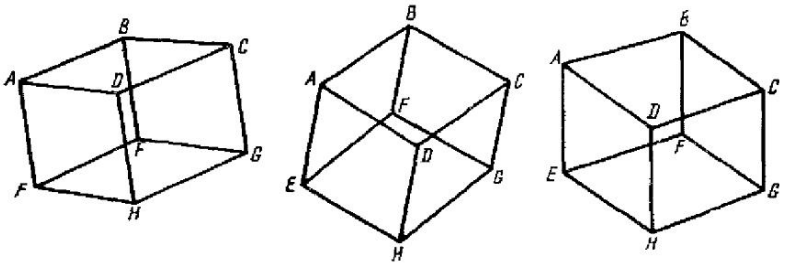


Рис 179 Определение межкадровой однородности объекта по наблюдаемости; на всех трех кадрах грани *ABCD, ADHE* и *DHGC* характеризуются одной и той же наблюдаемостью

объект — выпуклый, то нетрудно определить однородность объекта по наблюдаемости или межкадровую однородность по наблюдаемости. Обращаясь к приведенному примеру, допустим, что на первом кадре определена наблюдаемость вершины *D*. Пока вершина *D* не выходит за пределы многоугольника *ABCGHE*, наблюдаемость граней не изменяется. В разд. 17.8 приведены сведения о подходах к решению этой задачи, предложенных в последнее время.

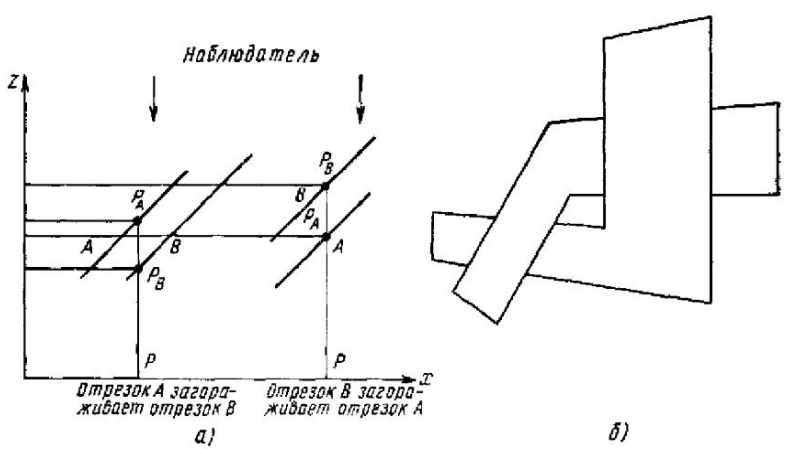


Рис. 17.10. Примеры нетривиальных случаев разделения видимых и невидимых элементов сцены:
а — для установления наблюдаемости недостаточно определения экстремальных значений по координате *z*, необходимо сопоставление точек P_A и P_B, *б* — две плоские поверхности взаимно загораживают друг друга

Следует отметить, что в нетривиальных случаях многие простые методы решения задачи разделения видимых и невидимых элементов сцены не срабатывают Так, на рис 17.10,а приведен пример случая, когда учет экстремального значения координаты z оказывается недостаточным. Если P — точка, принадлежащая пересечению проекций двух отрезков прямых, то для определения наблюдаемости достаточно сравнить лишь точки P_A и P_B, проектирующиеся в точку P. Этот метод можно использовать и в тех случаях, когда обе грани являются выпуклыми плоскими областями. Если же поверхности плоские, но невыпуклые области, то анализ случаев типа, приведенного на рис. 17.10,*б*, должен носить более детальный характер. Объем вычислений, необходимый для обработки неплоских поверхностей, еще больше, чем в случае плоскостей. Упростить задачу можно с помощью разбиения конечных участков поверхности, ограниченных замкнутыми кривыми, на меньшие конечные участки.

17.6. ОПИСАНИЕ НЕЛИНЕЙНЫХ ОБЪЕКТОВ

Алгоритм 17.2 имеет достаточно общий характер, и для его применения не имеет значения форма поверхности граней, если отвлечься от подробностей вычислительных процедур. Этот алгоритм предусматривает выполнение следующих операций: а) определение наличия пересечения плоскости, задаваемой уравнением y = постоянная, с гранью; б) если оказывается, что пересечение имеет место, определяется получаемая фигура; в) поиск ближайшей кривой по оси z. Это очень простые операции, если грани представляют собой плоские выпуклые многоугольники. Действительно, для выполнения операции а) необходимо лишь определить экстремальные значения координаты y на границах обрабатываемой грани. Операции б) и в) связаны исключительно с обработкой отрезков прямых. Если предполагается штриховка воспроизведенного изображения, необходимо также определить для каждого отрезка нормаль к поверхности (см. разд. 13.9). Для плоского многоугольника направление нормали фиксировано во все\ точках. Это упрощение перестает действовать в случае нелинейных конечных участков поверхности. Для того чтобы установить, пересекается ли рассматриваемый конечный участок поверхности с плоскостью, необходимо учитывать не только границу этого участка, но и *силуэт* его проекции — это необходимо для учета выступающих выпуклостей по верхности. На рис. 17.11 представлены два таких силуэта сферических конечных участков поверхности. Поскольку в этой си-

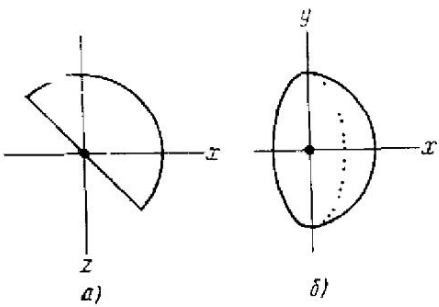


Рис 17.11 Силуэты конечных участков сферических поверхностей

туации направление нормали к поверхности не фиксировано, обрабатываться должны все внутренние точки поверхности.

Пусть $S\ (u,\ v)$—параметрическое уравнение поверхности и пусть кривые $S(u,\ 0)$, $S(u,\ 1)$, $S\ (0,\ v)$ и $S(1,\ v)$ ограничивают один из ее участков. Такое представление допускают поверхности Кунса (см. разд. 13.6), поверхности Безье (см. подразд. 13.7.1) и B-сплайн-поверхности (см. подразд. 13.7.2). Напомним, что $S(u,\ 0)$—заданная кривая в случае поверхностей Кунса [см. (13.18')], а в двух других — многочлен Безье [см. (13.31)] и B-сплайн [см. (13.33)] соответственно. То же относится и к остальным граничным кривым. Рассмотрим точку $S(u,\ v)$ с координатами $X(u,\ v)$, $Y(u,\ v)$ и $Z(u,\ v)$. Для того чтобы определить пересечение поверхности с заданной плоскостью y-c, необходимо решить уравнение

$$Y\ (u,\ v) = c. \tag{17.4}$$

В общем случае это уравнение задает зависимость между u и v, если $v\ (u)$ — выражение, представляющее величину v через величину u, то $S\ (u,\ v(u))$ задает кривую на плоскости $y = c$. Использование этого общего подхода непрактично и вместо этого можно попытаться получить численное решение следующих четырех уравнений:

$$Y\ (u,\ 0) = c, \tag{17.5а}$$

$$Y\ (u,\ 1) = c, \tag{17.5б}$$

$$Y\ (0,\ v) = c, \tag{17.5в}$$

$$Y\ (1,\ v) = c. \tag{17.5г}$$

Решение этих уравнений определяют четыре точки, в которых границы конечного участка поверхности пересекаются плоскостью, задаваемой уравнением $y = c$. Если эти уравнения не имеют решения, то это не означает, что плоскость не пересекается с данным участком поверхности, поскольку пересечение может иметь место во внутренней части участка. Силуэт поверхности, спроектированной на плоскость, перпендикулярную координатной оси z, образован теми точками, в которых касательная плоскость к поверхности параллельна координатной оси z. Следовательно, нормаль к поверхности не должна иметь компоненту, параллельную этой оси Согласно уравнению (13.5в) такие точки должны удовлетворять уравнению

$$X_u\ Y_v - Y_u\ X_v = 0, \tag{17.6}$$

где нижние индексы обозначают переменные, по которым берутся частные производные Кроме того, эти точки должны удовлетворять уравнению (174) Если необходимо определить диапазон значений координаты y для конечного участка поверхности, следует определить экстремальные значения по оси y. Последним соответствуют либо точки, находящиеся на границе, либо участ-

ки, на которых изменение значения $S(u, v)$ по оси y равно нулю, а именно:

$$\frac{\partial Y}{\partial u}(u, v) = 0, \quad \frac{\partial Y}{\partial v}(u, v) = 0. \tag{17.7}$$

Для выполнения штриховки требуется определить точные координаты, соответствующие определенному значению x, т. е. необходимо решить систему уравнений, включающую уравнение (17.6) и уравнение

$$X(u, v) = x. \tag{17.8}$$

Все эти уравнения — нелинейные, и для их решения обычно используются численные методы, предусматривающие предварительный выбор некоторого решения и его уточнение при помощи итерационных процедур [1]. В этом случае чрезвычайно важное значение приобретает однородность строк развертки по наблюдаемости, так как решения, полученные для некоторой строки, можно использовать в качестве предварительной оценки решения для следующей строки. Аналитическое решение оказывается возможным лишь в очень простых случаях. Приведем один из них в качестве примера, иллюстрирующего не столько метод решения, сколько используемые понятия.

Пример 17.1. Рассмотрим шаровой сектор, задаваемый следующими урав-

$$X(u, v) = \cos u \cos v, \quad Y(u, v) = \sin v, \quad Z(u, v) = \sin u \cos v, \tag{17.9a}$$

$$x + z \geqslant 0. \tag{17.9б}$$

Уравнение (17.9а), записанное в непараметрической форме, эквивалентно уравнению

$$x^2 + y^2 + z^2 = 1. \tag{17.9а'}$$

нениями:

Проекция этого шарового сектора на плоскость x-z имеет вид, приведенный на рис. 17.11,*а*, и состоит из точек, удовлетворяющих уравнению (17.9*б*) и неравенству

$$x^2 + z^2 \leqslant 1. \tag{17.10}$$

Сложнее определить проекцию шарового сектора на плоскость x-y. В этом случае граница конечного участка поверхности задается не приведенными выше

$$x + z = 0. \tag{17.10'}$$

Исключив переменную z из уравнений (17.9а') — (17.10'), устанавливаем, что проекция границы представляет собой эллипс, определяемый уравнением

$$2x^2 + y^2 = 1. \tag{17.11}$$

Если используется параметрическая форма, то справедливо соотношение

$$x + z = \sqrt{2} \sin(u + \pi/4) \cos v.$$

значениями и и v, *а* определяется как пересечение сферы и плоскости:

Правая часть равна нулю при $v = \pi / 2$ *или* $u = -\pi / 4$. В первом случае $x = z = 0$ и $y = 1$, а во втором $x = \cos v/\mathrm{sqrt}(2)$, $z = -\cos v/\mathrm{sqrt}(2)$, $y = \sin v$. Мы получили параметрическое уравнение эллипса для трехмерного случая. Для получения его

[1] Этот подход предложен Блинном [13.1], применившим для определения решения метод Ньютона. Суть этого метода изложена в статье [17.10].

проекции на плоскость *x-y* необходимо просто не учитывать значение *z*. Читатель может самостоятельно убедиться в том, что такая проекция эквивалентна уравнению (17.11). Любому фиксированному значению *y* соответствует некоторая пара значений *x*, определяемая из выражения $\pm sqrt(1-y^2)/2)$. Однако только одно из них соответствует границе проекции. Для того чтобы найти силуэт, решаются уравнения (17.7), что дает в результате значение $-\sin u \cos^2 v$, которое равно нулю при $u=0$ или $v = \pi/2$. Обращаясь к непараметрическому уравнению, находим, что z=0. Итак, силуэт представляет собой окружность $x^2+y^2=1$, следовательно, полная проекция состоит из одной дуги эллипса и одной дуги окружности, что и показано на рис. 17.11,*б*.

Альтернативой решению нелинейных уравнений служит последовательное разбиение конечных участков поверхности, продолжаемое до тех пор, пока они не приобретут размеры, позволяющие использовать для их аппроксимации многоугольники. В основе этого подхода лежит теорема 10.1, утверждающая, что область определения любого многочлена Безье можно разделить пополам, причем каждое из возникающих в результате подмножеств кривых также является многочленом Безье. Аналогичные результаты можно получить для кривых других типов.

17.7. ПРИДАНИЕ ВОСПРОИЗВОДИМОМУ ИЗОБРАЖЕНИЮ ЕСТЕСТВЕННОГО ВИДА

Для придания воспроизводимым изображениям вида естественной сцены использовались текстуры, рандомизация и многократное отражение света. Другое направление попыток, посвященных достижению этой же цели, связано с устранением «зубчатого» или «лестничного» характера изображений, полученных посредством кусочно-постоянного восстановления (см. разд. 2.4). В этом разделе будут кратко рассмотрены методы, используемые для придания изображению текстурированного вида.

Как отмечалось в разд. 3.3, текстура соответствует статистикам второго порядка, характеризующим распределение яркости на изображении. Чтобы некоторое синтезированное изображение давало зрительное впечатление текстуры, необходимо присваивать пикселу значение цвета или уровня яркости, исходя из цвета и яркости соседних с ним пикселов. Делать это можно двумя способами. Один из них предусматривает фиксацию геометрии поверхности, определяемой $S(u, v)$, и изменение цвета в соответствии со значениями функции $C(u, v)$. Значения функции $C(u, v)$ могут определяться с помощью некоторой формулы, либо она может быть табулирована, но в то же время их можно порождать с помощью некоторого случайного процесса. Более реалистический подход предусматривает изменения формы поверхности, производимые, например, введением выпуклостей и складок, и создание текстурного эффекта за счет вариаций отраженного света. Один из способов моделирования такой грубой поверхности заключается в возмущении поверхности по направлению ее нормали

при помощи заданной (скалярной) функции $P(u, v)$. В таком случае:

$$X^p(u, v) = X(u, v) + P(u, v)\frac{F_x}{\|F\|}, \tag{17.12а}$$

$$Y^p(u, v) = Y(u, v) + P(u, v)\frac{F_y}{\|F\|}, \tag{17.12б}$$

$$Z^p(u, v) = Z(u, v) + P(u, v)\frac{F_z}{\|F\|}, \tag{17.12в}$$

где XP, YP и ZP — возмущенные значения поверхности, (F_x, F_y, F_z) — компоненты нормали к поверхности, определяемой уравнениями (13.5), и $\|F\|$ — ее норма. Для определения кажущейся яркости следует найти значения нормали к поверхности, подвергаемой возмущению. Обозначим через X^P_U частную производную возмущенного значения X^P по u. Аналогичные обозначения используются и для остальных производных. В таком случае получаем, что

$$X^p_u = X_u + P_u\frac{F_x}{\|F\|} + P\frac{\partial}{\partial u}\left[\frac{F_x}{\|F\|}\right]. \tag{17.13}$$

Теперь допустим, что функция $P(u, v)$ принимает очень небольшие значения, но имеет большие производные. Тогда можно пренебречь третьим членом в правой части уравнения (17.13). Аналогичные выражения можно получить для производных возмущенных значений Y^P_U, Z^P_U и т. д. Следовательно, уравнение для нормали возмущенной поверхности имеет вид

$$F^p_x = Y^p_u Z^p_u - Z^p_u Y^p_v = \left[Y_u - P_u\frac{F_y}{\|F\|}\right]\left[Z_v + P_v\frac{F_z}{\|F\|}\right] -$$

$$- \left[Z_u + P_u\frac{F_z}{\|F\|}\right]\left[Y_v + P_v\frac{F_y}{\|F\|}\right]. \tag{17.14}$$

Перемножив члены, заключенные в скобки, использовав уравнение (13.5а) и проведя сокращения, получаем

$$F^p_x = F_x + (P_u Z_v - Z_u P_v)\frac{F_y}{\|F\|} + (Y_u P_v - P_u Y_v)\frac{F_z}{\|F\|}. \tag{17.15}$$

Повторив эти преобразования для двух других компонент нормали, получим аналогичные уравнения. Это означает, что нормаль к возмущенной поверхности можно определить по нормали к невозмущенной поверхности при помощи умножения последней на симметричную матрицу 3X3. Главная диагональ матрицы состоит из единиц, а выражения для остальных ее элементов можно получить из уравнения (17.15).

Функцию $P(u, v)$ можно задать с помощью таблицы, в явном виде или посредством ее отождествления с некоторой случайной величиной. Более подробные сведения по этому поводу можно найти в диссертации [13.1].

Если возмущение поверхности или ее цвета в некоторой точке не зависит от возмущений в других точках, то рассматриваемая

задача очень сильно упрощается. Единственное, что в этом случае для ее решения требуется — обратиться к некоторой процедуре порождения (псевдо) случайных чисел при обработке каждой точки поверхности. Использование подобных независимых возмущений не позволяет получить реалистическую модель текстурированных поверхностей материальных объектов. Не так уж трудно ввести корреляцию между отдельными точками поверхности, если возмущающая функция $P(u, v)$ задается достаточно большой таблицей. Тем не менее это нельзя сделать «на лету», так как точки, которые кажутся расположенными близко в процессе воспроизведения изображения, могут очень значительно отстоять друг от друга на поверхности. Более подробные сведения по этому поводу, а также относительно аналогичных методов можно найти в разд. 17.8.

17.8. БИБЛИОГРАФИЧЕСКИЕ ЗАМЕЧАНИЯ

Одним из первых обзоров, посвященных алгоритмам разделения видимых и невидимых элементов сцены, является работа [17.13]. Описанный в этой главе алгоритм, основанный на использовании тетрарного дерева, является модификацией алгоритма, предложенного Уорноком (точную ссылку на его работу можно найти в монографии [1.4] или статье [17.13]); отметим, что это первое упоминание в научной печати о применении такой структуры данных в обработке изображений. Алгоритм построчного разделения видимых и невидимых элементов сцены основан на алгоритме, предложенном Уоткинсом (см. монографию [1.4] или статью [17.13]). Ряд методов, предложенных в более поздние годы, описан в статье [17.10], в том числе решение задачи разделения видимых и невидимых элементов сцены для нелинейных конечных участков поверхности. В статье [17.14] рассмотрена комбинированная задача, включающая разделение видимых и невидимых элементов сцены и штриховку, в том числе для случая многократного отражения света. В диссертации [13.1] проведено тщательное исследование растрового алгоритма построчного разделения видимых и невидимых элементов сцены для случая, когда объекты описываются с помощью конечных участков поверхности. В ней также подробно рассмотрены методы получения текстурообразных изображений и наши разд. 17.6 и 17.7 в значительной мере основаны на материалах этой работы. (Резюме части результатов этой диссертации имеется в статье [17.10].) В статье [17.3] описывается один из методов рандомизации, который был успешно использован для получения изображений участков земной поверхности.

В статье [17.8] рассматривается алгоритм разделения видимых и невидимых элементов сцены, работающий в пространстве объектов, причем объектами служат шары и цилиндры. В алгоритме используется функция, аналогичная процедуре *LOC*, рассмотренной в разд. 17.3, однако вместо сравнения граней и фраг-

ментов изображения она предусматривает сравнение граней между собой. В докладе [17.11] излагается развитие этой работы, позволяющее использовать штриховку (включая светлые участки изображения). С помощью предложенного метода можно преодолеть ряд трудностей, связанных с нелинейными конечными участками поверхности, опираясь на преимущества, обеспечиваемые простотой геометрии шара. Методы, описанные в этих работах, были использованы для получения изображений моделей молекул. Еще один пример применения сфер в машинной графике описан в статье [17.1]. Модель материального объекта получена не на основе использования конечных участков поверхности, а как оболочка пересекающихся шаров. Авторы предложили модель тела человека, построенную из примерно трехсот шаров; они указывают, что для получения соответствующего описания потребовалось бы три тысячи конечных участков поверхности. Сходная работа представлена в статье [17.12], а развитие этого метода дано в [17.9].

В последних исследованиях проблемы наблюдаемости объектов сцены значительное внимание уделялось определению порядка сопоставления граней и фрагментов изображения. В статье [17.2] обсуждаются теоретические аспекты подхода, основанного на использовании пространства объектов. В докладе [17.4] изложен алгоритм того же типа, что и алгоритм 17.1, однако оснащенный тщательно разработанной стратегией определения пересечений фрагмента изображения и грани. В результате обеспечен переход от квадратичной временной сложности к линейной. В докладе [17.6] рассматривается комбинация межкадровой однородности по наблюдаемости и однородности объекта по наблюдаемости.

В сборнике [17.5] помещены статьи, посвященные проблеме восприятия; они могут оказаться полезными для специалистов в области конструирования дисплеев.

17.9. ЗАДАЧИ

17.1. Определите форму трехмерного объекта, который после применения к нему преобразования (16.20) превратился в изображение шара.

17.2. Напишите программу, реализующую алгоритм 17.1. Для упрощения задачи разрезания можно считать, что все грани и правильные прямоугольники параллельны плоскости *x-y*.

17.3. Реализуйте процедуру *LOC(W, F),* описанную в разд. 17.3. Заметьте, что можно опустить те части алгоритма разрезания, которые определяют пересечения прямых. Сократите процедуру проверки посредством обращения к вызывающей программе при каждом обнаружении пары пересекающихся сторон. С другой стороны, необходимо отнестись внимательно к разграничению случаев, когда *F* покрывает *W*, *W* покрывает *F*, а *F* и *W* разнесены друг от друга.

17.4. Реализуйте алгоритм 17.2 с учетом условия, введенного в задаче 17.2.

17.5. Давайте сделаем кинофильм! В простейшем варианте можно использовать два объекта — куб и правильный четырехгранник. Длина ребра последнего должна составлять примерно половину длины ребра куба. Куб следует изобразить таким образом, чтобы одна его диагональ была расположена вертикально, и он должен поворачиваться вокруг этой диагонали. (Правильный выбор системы координат существенно упростит формулы для преобразований

поворота.) Четырехгранник следует разместить по отношению к кубу на расстоянии, равном утроенной длине ребра последнего; он также должен поворачиваться относительно такой же оси. Угловая скорость четырехгранника должна составлять около одной десятой угловой скорости куба. Напишите программу для получения последовательности изображений этой сцены. В целом сцена должна напоминать изображение планеты, вращающейся вокруг Солнца. Это задание можно усложнить, введя в него дополнительные объекты или увеличив число граней многогранников.

Указание. Это пример ситуации, в которой при решении задачи разделения видимых и невидимых объектов имеет смысл использовать однородность объектов по наблюдаемости [1]. Большая часть пикселов относится к фону и большую часть времени их проекции будут разнесены на изображении.

17.6. Повторите решение предыдущей задачи, заменив многогранники шарами. Поскольку при этой замене решение задачи разделения видимых и невидимых элементов сцены упрощается, введите следующие усложнения. Добавьте планеты, плоскости орбит которых почти нормальны плоскости изображения, считайте, что неподвижный объект является источником света и имеет место затенение поверхностей объектов.

[1] Когда это задание было предложено студентам Принстонского университета, около половины из них воспользовались однородностью объекта по наблюдаемости, а остальные применяли общий алгоритм в пространстве изображений. На выполнение задания студенты затратили приблизительно три недели.

[2] Номера алгоритмов соответствуют их нумерации в тексте. — *Прим. перев.*