**Тема 3. Компьютерный анализ данных**

**Лекция 10. Методы и алгоритмы обработки и анализа данных**

**1. Общие методы и алгоритмы обработки и анализа данных.**

Виды анализа:

- системный (типология, классификация);
- структурный (структуры и их связи);
- событийный (события).

Разделение видов анализа по противоположным признакам:

- статический и динамический;
- детерминистический и стохастический.

**1.1. Статистическая проверка гипотез**. Приводится общая схема проверки гипотез и на её основе решаются задачи проверки гипотез: о математическом ожидании, дисперсиях, равенстве математических ожиданий, выявлении аномальных измерений, об однородности ряда дисперсий, согласованности выбранного закона распределения и гистограммы.

**1.2. Классификация в распознавании образов**. Строится общая схема системы распознавания и обсуждается идея классификации. Подходы: байесовская теория принятия решений, прямые методы восстановления решающих функций, схема персептрона (нейросетевые методы).

**1.3. Планирование эксперимента**. Рассматриваются: планирование эксперимента при построении линейной статической модели объекта, метод крутого восхождения по поверхности отклика, полный факторный эксперимент первого порядка, дробные реплики, насыщенные планы, устранение кусочно-постоянного дрейфа за счет разбиения матрицы планирования на блоки, алгоритмы обработки результатов эксперимента, ортогональные и ротатабельные планы второго порядка, метод случайного баланса.

**1.4. Методы непараметрической обработки информации.** Излагается общая идея оценивания статистических характеристик стохастических объектов, представленных в виде функционалов от плотностей распределения вероятностей. Строятся оценки для функции и плотности распределения (простейшие, полиграммы, *k* ближайших соседей, Розенблатта – Парзена) и приводятся их свойства. На основе оценок Розенблатта – Парзена получены состоятельные оценки: моментов случайных величин, энтропии, условной энтропии, условной плотности распределения, регрессии, средней условной энтропии, среднего количества информации, дисперсионных характеристик. Синтезируются робастные оценки регрессии. На основе использования непараметрических оценок ***инверсных регрессий*** строятся алгоритмы: адаптивного управления при априорной неопределенности, оптимального управления, управления экстремальными объектами, минимизации функций, классификации в распознавании образов.

**1.5. Методы экспериментальной оптимизации.** Рассматриваются: методы одномерного поиска минимума унимодальных функций, метод одномерного глобального поиска, последовательный симплексный метод, метод деформируемого многогранника, градиентный алгоритм с использованием ортогонального планирования первого порядка, алгоритм Ньютона с использованием планирования второго порядка, методы случайного поиска, новый метод (усреднения координат) **глобальной** оптимизации недифференцируемых стохастических функций.

**1.6. Идентификация статических моделей.** Дается общая постановка задачи подстройки параметров нелинейных моделей. Анализируются различные варианты критерия наименьших квадратов. Рассматриваются алгоритмы метода наименьших квадратов при линейной параметризации модели и исследуются свойства полученных параметров. На основе метода последовательной линеаризации строятся алгоритмы расчета параметров нелинейных моделей с использованием квадратичных критериев и спектра критериев, обеспечивающих получение оценок робастных по отношению к выбросам помех. Рассматривается система алгоритмов адаптивной обработки информации на основе критериев наименьших квадратов, робастных критериев, при частичном забывании старых экспериментальных данных, при линейных и нелинейных моделях, стационарных и нестационарных параметрах моделей. Структура рекуррентных алгоритмов для всех вышеуказанных случаев сохраняет свой вид, в них меняются лишь отдельные элементы. В заключение этого раздела построен и исследован простейший адаптивный алгоритм перестройки параметров линейных и нелинейных моделей. Проведены обобщения алгоритма на более сложные векторно-матричные модели. Изложена идея многоэтапного метода селекции при построении моделей сложных объектов.

**1.7. Идентификация динамических моделей объектов.** Рассматриваются вопросы синтеза оптимальной структуры дискретных динамических моделей стохастических объектов. Строятся алгоритмы подстройки параметров моделей с использованием как функций чувствительности, так и итеративных моделей. Рассматриваются простейший адаптивный алгоритм и модифицированный алгоритм наименьших квадратов. Построенные алгоритмы применяются в следующем разделе при синтезе устройств адаптивного управления стохастическими объектами.

**1.8. Адаптивное управление с идентификацией.** Дается постановка задачи адаптивного управления динамическими стохастическими объектами и анализируются основные подходы к синтезу алгоритмов управления. За основу рассмотрения выбран подход, основанный на построении (идентификации) моделей прогноза выхода систем. На простых примерах проводится синтез алгоритмов управления, строятся структурные схемы устройств управления, вычисляется ошибка работы систем. Метод синтеза обобщается на основные классы динамических систем: линейные, нелинейные, с чистыми запаздываниями и без них.

## 2. Алгоритмы сортировки

### 2.1. Сортировка пузырьком / Bubble sort

Будем идти по массиву слева направо. Если текущий элемент больше следующего, меняем их местами. Делаем так, пока массив не будет отсортирован. Заметим, что после первой итерации самый большой элемент будет находиться в конце массива, на правильном месте. После двух итераций на правильном месте будут стоять два наибольших элемента, и так далее. Очевидно, не более чем после n итераций массив будет отсортирован. Таким образом, асимптотика в худшем и среднем случае - $O\left(n^2\right)$, в лучшем случае - $O\left(n\right)$.

### 2.2. Шейкерная сортировка / Shaker sort

Шейкерная (сортировка перемешиванием или коктейльная сортировка) исправляет некоторые дефекты сортировки пузырьком. Для последней известно, что она работает медленно на тестах, в которых маленькие элементы стоят в конце (их еще называют <черепахами>). Такой элемент на каждом шаге алгоритма будет сдвигаться всего на одну позицию влево. Поэтому будем идти не только слева направо, но и справа налево. Будем поддерживать два указателя begin и end, обозначающих, какой отрезок массива еще не отсортирован. На очередной итерации при достижении end вычитаем из него единицу и движемся справа налево, аналогично, при достижении begin прибавляем единицу и двигаемся слева направо. Асимптотика у алгоритма такая же, как и у сортировки пузырьком, однако реальное время работы лучше.

### 2.3. Сортировка расческой / Comb sort

Еще одна модификация сортировки пузырьком. Для того, чтобы избавиться от <черепах>, будем переставлять элементы, стоящие на расстоянии. Зафиксируем некоторое расстояние, и будем идти слева направо, сравнивая элементы, стоящие на этом расстоянии, переставляя их, если необходимо. Очевидно, это позволит <черепахам> быстро добраться в начало массива. Оптимально изначально взять расстояние равным длине массива, а далее делить его на некоторый коэффициент, равный примерно 1.247. Когда расстояние станет равно единице, выполняется сортировка пузырьком. В лучшем случае асимптотика равна $O\left(n\log n\right)$, в худшем - $O\left(n^2\right)$.

### 2.4. Сортировка вставками / Insertion sort

Создадим массив, в котором после завершения алгоритма будет лежать ответ. Будем поочередно вставлять элементы из исходного массива так, чтобы элементы в массиве-ответе всегда были отсортированы. Асимптотика в среднем и худшем случае - $O\left(n^2\right)$, в лучшем - $O\left(n\right)$. Реализовывать алгоритм удобнее по-другому (создавать новый массив и реально что-то вставлять в него относительно сложно): просто сделаем так, чтобы отсортирован был некоторый префикс исходного массива, вместо вставки будем менять текущий элемент с предыдущим, пока они стоят в неправильном порядке.

### 2.5. Сортировка Шелла / Shellsort

Используем ту же идею, что и сортировка расческой, и применим к сортировке вставками. Зафиксируем некоторое расстояние. Тогда элементы массива разобьются на классы - в один класс попадают элементы, расстояние между которыми кратно зафиксированному расстоянию. Отсортируем сортировкой вставками каждый класс. В отличие от сортировки расческой, неизвестен оптимальный набор расстояний. Существует довольно много последовательностей с разными оценками.

Последовательность Шелла - первый элемент равен длине массива, каждый следующий вдвое меньше предыдущего. Асимптотика в худшем случае - $O(n^2)$. Последовательность Хиббарда - 2n-1, асимптотика в худшем случае - $O(n\sqrt{n})$, последовательность Седжвика - $O(n^{4/3})$, последовательность Пратта (все произведения степеней двойки и тройки) - $O(n\log n)$. Все эти последовательности нужно рассчитать только до размера массива и запускать от большего от меньшему (иначе получится просто сортировка вставками).

**2.6. Сортировка деревом / Tree sort**

Будем вставлять элементы в двоичное дерево поиска. После того, как все элементы вставлены достаточно обойти дерево в глубину и получить отсортированный массив. Если использовать сбалансированное дерево, например красно-черное, асимптотика будет равна $O(n\log n)$ в худшем, среднем и лучшем случае. В реализации использован контейнер multiset.

**2.7. Гномья сортировка / Gnome sort**

Алгоритм похож на сортировку вставками. Поддерживаем указатель на текущий элемент, если он больше предыдущего или он первый - смещаем указатель на позицию вправо, иначе меняем текущий и предыдущий элементы местами и смещаемся влево.

**2.8. Сортировка выбором / Selection sort**

На очередной итерации будем находить минимум в массиве после текущего элемента и менять его с ним, если надо. Таким образом, после i-ой итерации первые i элементов будут стоять на своих местах. Асимптотика: $O(n^2)$ в лучшем, среднем и худшем случае. Нужно отметить, что эту сортировку можно реализовать двумя способами - сохраняя минимум и его индекс или просто переставляя текущий элемент с рассматриваемым, если они стоят в неправильном порядке. Первый способ оказался немного быстрее, поэтому он и реализован.

**2.9. Пирамидальная сортировка / Heapsort**

Развитие идеи сортировки выбором. Воспользуемся структурой данных <куча> (или <пирамида>, откуда и название алгоритма). Она позволяет получать минимум за O(1), добавляя элементы и извлекая минимум за $O(n\log n)$. Таким образом, асимптотика $O(n\log n)$ в худшем, среднем и лучшем случае.

**2.10. Быстрая сортировка / Quicksort**

Выберем некоторый опорный элемент. После этого перекинем все элементы, меньшие его, налево, а большие - направо. Рекурсивно вызовемся от каждой из частей. В итоге получим отсортированный массив, так как каждый элемент меньше опорного стоял раньше каждого большего опорного. Асимптотика: $O(n\log n)$ в среднем и лучшем случае, $O(n^2)$. Наихудшая оценка достигается при неудачном выборе опорного элемента. Стандартная реализация – идем одновременно слева и справа, находим пару элементов, таких, что левый элемент больше опорного, а правый меньше, и меняем их местами.

**2.11. Сортировка слиянием / Merge sort**

Сортировка, основанная на парадигме <разделяй и властвуй>. Разделим массив пополам, рекурсивно отсортируем части, после чего выполним процедуру слияния: поддерживаем два указателя, один на текущий элемент первой части, второй - на текущий элемент второй части. Из этих двух элементов выбираем минимальный, вставляем в ответ и сдвигаем указатель, соответствующий минимуму. Слияние работает за $O(n)$, уровней всего $O(\log n)$, поэтому асимптотика $O(n\log n)$. Эффективно заранее создать временный массив и передать его в качестве аргумента функции. Эта сортировка рекурсивна, как и быстрая, а потому возможен переход на квадратичную при небольшом числе элементов.

**2.12. Сортировка подсчетом / Counting sort**

Создадим массив размера $r$ - l, где l - минимальный, а $r$ - максимальный элемент массива. После этого пройдем по массиву и подсчитаем количество вхождений каждого элемента. Теперь можно пройти по массиву значений и выписать каждое число столько раз, сколько нужно. Асимптотика - $O(n+r-1)$. Можно модифицировать этот алгоритм, чтобы он стал стабильным: для этого определим место, где должно стоять очередное число (это просто префиксные суммы в массиве значений) и будем

идти по исходному массиву слева направо, ставя элемент на правильное место и увеличивая позицию на 1.

**2.13. Блочная сортировка / Bucket sort**

Блочная (корзинная или карманная) сортировка состоит в следующем. Пусть *l* - минимальный, а *r* - максимальный элемент массива. Разобьем элементы на блоки, в первом будут элементы от *l* до *l* + *k*, во втором - от *l* + *k* до *l* + 2*k* и т.д., где *k* = (*r* - l) / количество блоков. Если количество блоков равно двум, то данный алгоритм превращается в разновидность быстрой сортировки. Асимптотика этого алгоритма неясна, время работы зависит и от входных данных, и от количества блоков. Утверждается, что на удачных данных время работы линейно. Реализация этого алгоритма оказалась одной из самых трудных задач. Можно сделать это так: просто создавать новые массивы, рекурсивно их сортировать и склеивать. Однако такой подход все же довольно медленный.

**2.14. Поразрядная сортировка / Radix sort**

Поразрядная (цифровая) сортировка имеет две версии.

**2.14.1. LSD (least significant digit)**

Представим каждое число в двоичном виде. На каждом шаге алгоритма будем сортировать числа таким образом, чтобы они были отсортированы по первым k * i битам, где k - некоторая константа. Из данного определения следует, что на каждом шаге достаточно стабильно сортировать элементы по новым k битам. Для этого идеально подходит сортировка подсчетом (необходимо 2k памяти и времени, что немного при удачном выборе константы). Асимптотика: $O(n)$, если считать, что числа фиксированного размера (в противном случае нельзя считать, что сравнение двух чисел выполняется за единицу времени). Реализация довольно проста.

**2.14.2. MSD (most significant digit)**

Это разновидность блочной сортировки. В один блок будут попадать числа с равными k битами. Асимптотика такая же, как и у LSD версии. Реализация похожа на блочную сортировку, но проще. В ней используется функция digit, определенная в реализации LSD версии.

**2.15. Битонная сортировка / Bitonic sort**

Идея алгоритма – исходный массив преобразуется в битонную последовательность - последовательность, которая сначала возрастает, а потом убывает. Ее можно эффективно отсортировать следующим образом: разобьем массив на две части, создадим два массива, в первый добавим все элементы, равные минимуму из соответственных элементов каждой из двух частей, а во второй - равные максимуму. Утверждается, что получатся две битонные последовательности, каждую из которых можно рекурсивно отсортировать тем же образом, после чего можно склеить два массива (так как любой элемент первого меньше или равен любого элемента второго). Для того, чтобы преобразовать исходный массив в битонную последовательность, сделаем следующее: если массив состоит из двух элементов, можно просто завершиться, иначе разделим массив пополам, рекурсивно вызовем от половинок алгоритм, после чего отсортируем первую часть по порядку, вторую в обратном порядке и склеим. Очевидно, получится битонная последовательность. Асимптотика: $O(n\log n)$. P размер массива должен быть равен степени двойки.

**2.16 Timsort**

Гибридная сортировка, совмещающая сортировку вставками и сортировку слиянием. Разобьем элементы массива на несколько подмассивов небольшого размера, при этом будем расширять подмассив, пока элементы в нем отсортированы. Отсортируем подмассивы сортировкой вставками, пользуясь тем, что она эффективно работает на отсортированных массивах. Далее будем сливать подмассивы как в сортировке слиянием, беря их примерно равного размера (иначе время работы приблизится к квадратичному). Для этого удобного хранить подмассивы в стеке, поддерживая инвариант - чем дальше от вершины, тем больше размер, и сливать подмассивы на верхушке только тогда, когда размер третьего по отдаленности от вершины подмассива больше или равен сумме их размеров. Асимптотика: $O(n)$ в лучшем случае и $O(n\log n)$ в среднем и худшем случае. Реализация нетривиальна.

**2.17 Сортировка Бэтчера**

Сортировка Бэтчера – коммерческая сортировка. Предполагает использование параллельных вычислений.

**2.18 Резюме**

Для целых чисел имеет смысл рассматривать различные алгоритмы из класса $O(n) - O(n\log n)$. Для вещественных чисел – пирамидальная, битонная, сортировка слиянием.

**3.10. Алгоритмы поиска**

**3.10.1. Алгоритмы поиска в линейном массиве данных**

- двоичный координатный поиск
- двоичный координатный поиск с сортировкой
- применение алгоритмов п. 1.10.3.

**3.10.2. Алгоритмы поиска в многомерном массиве данных**

- m-мерный координатный поиск
- m-мерный координатный поиск с сортировкой
- применение алгоритмов п. 1.10.3

**3.10.3. Алгоритмы поиска на графах**

- поиск в глубину
- поиск в ширину
- лексикографический поиск
- поиск по первому лучшему совпадению
- двунаправленный поиск
- алгоритм Беллмана - Форда
- алгоритм Левита
- алгоритм Ли
- алгоритм Дейкстры
- информированный и неинформированный поиск.

**4. Алгоритмы оптимизации**

**5. Алгоритмы управления**

**6. Методы и алгоритмы обработки и анализа изображений.**

Материал из книги – Сойфер В.А. Методы компьютерной обработки изображений. – М.: Физматлит, 2003.

МАТЕМАТИКА

# КОМПЬЮТЕРНАЯ ОБРАБОТКА ИЗОБРАЖЕНИЙ.

ЧАСТЬ 1. МАТЕМАТИЧЕСКИЕ МОДЕЛИ

В.А. Сойфер

Самарский государственный аэрокосмический университет

Исходный URL: http://www.pereplet.ru/obrazovanie/stsoros/49.html

## ВВЕДЕНИЕ

Недаром говорят: "Лучше один раз увидеть, чем сто раз услышать". Исследования подтверждают, что информационная пропускная способность органов зрения значительно выше, чем у других каналов передачи информации, доступных человеку. В теории информации доказано, что, подбрасывая монету и наблюдая результат, мы всякий раз получаем одну двоичную единицу (бит) информации. Каждая буква в тексте несет примерно четыре бита информации. Изображение участка поверхности Земли, полученное из космоса, содержит примерно 10 миллионов бит информации! Переработать такое количество информации под силу только самому современному компьютеру. Чтобы научить машину обрабатывать изображения, требуется иметь мощный комплекс технических средств, математический аппарат, алгоритмы и большое количество программ.

Часть первая публикации посвящена построению математических моделей оптических изображений и их дискретным представлениям. Во второй части публикации рассматриваются методы и алгоритмы, а также несколько примеров решения прикладных задач.

## 1. МАТЕМАТИЧЕСКИЕ МОДЕЛИ ОПТИЧЕСКИХ ИЗОБРАЖЕНИЙ

### 1.1. Функция яркости

Необходимость построения математической модели возникает сразу же при использовании компьютера для обработки изображений. Оценивая "на глаз" расстояние между двумя предметами, мы не задумываемся о том, как это делается. Поручив это компьютеру, мы обязаны научить его выполнять подобные действия, то есть заложить в него соответствующие данные и алгоритмы. Хорошо известно, что компьютер имеет дело с массивами чисел в качестве данных. Таким образом, первой задачей компьютерной обработки изображений является перевод изображений в числовую форму. Это требует конкретизации самого понятия "изображение".

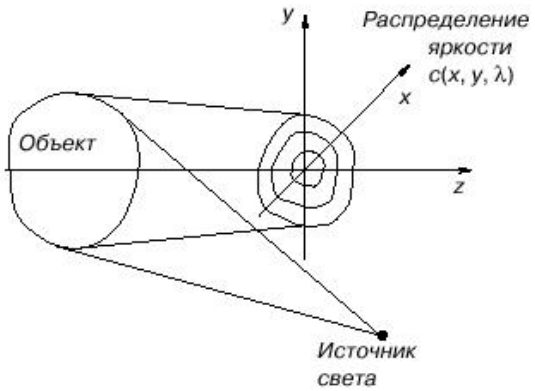



**Рис. 1.** Формирование изображения объекта, освещенного источником света.

Рассмотрим объект, освещенный источником света (рис. 1).
На некотором расстоянии от объекта распределение энергии источника светового излучения, отраженного объектом, по пространственным координатам x, y и по длинам волн l описывается функцией c(x, y, l). Эта функция является неотрицательной; ее максимальное значение в изображающих системах ограничено предельной величиной светочувствительности регистрирующих сред,

$0 < c(x, y, l) < A,$

где A - максимальная яркость изображения.

Геометрические размеры изображения ограничены характеристиками формирующей системы и параметрами фоторегистрирующей среды. Будем полагать, что все изображения отличны от нуля в прямоугольной области:

$-Lx < x < Lx$ , $-Ly < y < Ly$ .

Человеческое зрение и видеодатчики обладают спектральной чувствительностью, описываемой функцией u(l). Например, как известно, человеческий глаз обладает чувствительностью к свету в диапазоне волн от $\lambda_{min} = 0{,}35$ мкм до $\lambda_{max} = 0{,}78$ мкм. При этом функция спектральной чувствительности достигает своего максимума приблизительно в середине этого диапазона и спадает к его краям.

Каждый видеодатчик обладает индивидуальной характеристикой спектральной чувствительности, обусловленной физикой прибора. Имеются видеодатчики ультрафиолетового и инфракрасного диапазонов, которые широко используются, например, при проведении спектрозональных съемок Земли из космоса.

Как в случае наблюдения объекта человеком, так и в случае использования видеодатчика наблюдаемое изображение является результатом усреднения функции c(x, y, l) по диапазону длин волн с весовой функцией u(l) и описывается выражением

Функцию f(x, y) в дальнейшем будем называть изображением. Таким образом, изображение - это ограниченная функция двух пространственных переменных, заданная на ограниченной прямоугольной области.

**1.2. Двумерные линейные системы**

Из курса физики хорошо известно понятие оптической системы, осуществляющей преобразование изображений по определенным правилам, определяемым совокупностью используемых в ней оптических элементов и их взаимосвязью.

С математической точки зрения под системой будем понимать правило L, ставящее в соответствие входной функции f выходную функцию g. Различают одномерные 1D и двумерные 2D системы. Одномерные системы преобразуют функции одной переменной:

$g(x) = L[f(x)]$.

Соответственно двумерные системы преобразуют функции двух переменных:

$g(x, y) = L[f(x, y)]$.

Оптические системы по сути своей являются двумерными, но в некоторых случаях могут рассматриваться как одномерные.

Особое место среди всевозможных систем занимают линейные системы. Система называется линейной, если для нее справедлив принцип суперпозиции (наложения), который заключается в том, что отклик системы на взвешенную сумму двух входных воздействий равен взвешенной сумме откликов на каждое из воздействий, то есть

$L[a1\ f1(x, y) + a2\ f2(x, y)] = a1L[f(x, y)] + a2L[f2(x, y)]$.

Принцип суперпозиций можно выразить в более общем виде, рассматривая произвольное число M входных воздействий:

В изучении оптических систем фундаментальную роль играет понятие точечного источника света. Точечный источник света описывается дельта-функцией Дирака

Таким образом, точечный источник обладает бесконечно большой плотностью яркости в бесконечно малой пространственной области - в точке. Безусловно, это математическая абстракция, однако исключительно полезная в физике и допускающая ясную физическую трактовку: дельта-функция может быть определена как предел обычной функции, например

Согласно выражению (9) дельта-функция может рассматриваться как бесконечно узкая колоколообразная функция (рис. 2).

Можно также ввести дельта-функцию, расположенную не в начале координат, а в произвольной точке с координатами (u, u), по формуле

Дельта-функция обладает следующими важными свойствами:

1) Свойство нормировки:

Физически это означает, что, хотя плотность яркости точечного источника бесконечна, энергия его ограничена и равна единице.

2) Фильтрующее свойство

где f(x, y) - произвольная функция двух переменных. Интегралы в (11) и (12) берутся по бесконечно большой пространственной области D. Доказательства свойств 1) и 2) выполняются с помощью подстановки в (11) и (12) выражения (9) и раскрытия предела.

Рассмотрим 2D-линейную систему, на вход которой подан сигнал в виде дельта-функции. Реакция системы на дельта-функцию будет различной для различных систем, называется импульсным откликом и служит характеристикой 2D-системы. Систему называют пространственно-инвариантной, если ее импульсный отклик зависит от разности координат входной (x, y) и выходной (x, h) плоскостей. Для оптической системы, показанной на рис. 3, это означает, что при перемещении точечного источника во входной (предметной) области изображение этого предмета в плоскости наблюдения будет также изменять положение, но сохранять форму.

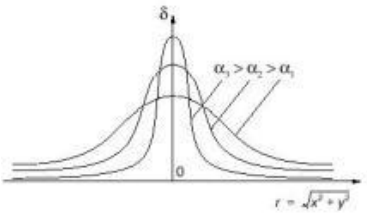



**Рис. 2.** Физическая трактовка дельта-функции Дирака.

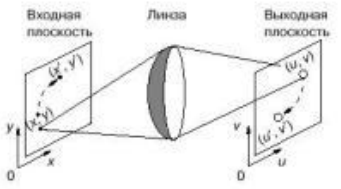



**Рис. 3.** Оптическая пространственно-инвариантная система.

Для пространственно-инвариантных систем импульсный отклик описывается функцией

h(x - u, y - u) Ï h(x, h),

где x = x - u, h = y - Яu,

h(x, h) Ï L[d(x, y)].

Используя функцию импульсного отклика, можно записать уравнение, связывающее изображения на входе и выходе 2D-линейной оптической системы. Для этого представим входной сигнал f(x, y) в виде (12) и подадим его на вход 2D-системы с характеристикой h(x, h). Выходной сигнал запишем в виде

Поскольку операция L линейна и операция интегрирования в фигурных скобках (15) также линейна, их можно поменять местами и записать

Учитывая, что по определению

L{d(x - x, y - h)} Ï h(x - x, y - h),

окончательно получим выражение, устанавливающее связь между изображениями во входной и выходной плоскостях линейной системы:

Уравнение (16) называется интегралом свертки. Из этого уравнения следует, что, зная импульсный отклик оптической системы h(x, h), можно рассчитать выходное изображение по входному.

Процесс свертки иллюстрирует рис. 4.

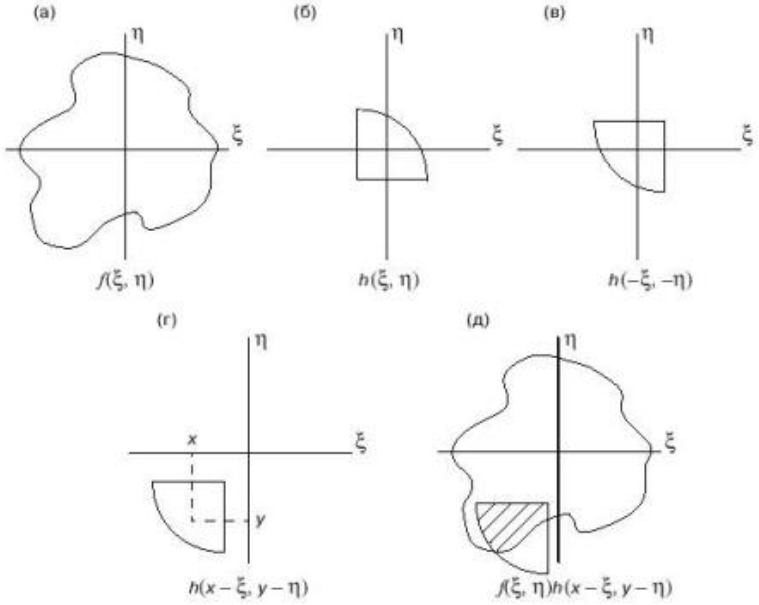



**Рис. 4.** Пример двумерной свертки.

На рис. 4а и 4б изображены функция f(x, y) на входе и импульсный отклик. На рис. 4а показан импульсный отклик при обращении координат, а на рис. 4г - со сдвигом на величину x, y. На рис. 4д заштрихована область, в которой произведение f(x, h) h(x - x, y - h), входящее в подынтегральное выражение (16), не равно нулю. Интегрирование по этой области дает величину g(x, y) для заданных значений координат x, y. Таким образом, функция g(x, y) на выходе может быть найдена сканированием входной функции скользящим "окном" - обращенным импульсным откликом, и интегрированием по области, в которой эти функции перекрываются.

### 1.3. Средства ввода изображений

Техническая задача, которую необходимо решить в компьютерной обработке изображений, - это ввод оптических изображений в память компьютера и вывод (визуализация) изображений. К счастью, в современных компьютерах задача визуализации решена. Для этих целей используются высокоразрешающие цветные дисплеи и другая техника отображения информации.

Ввод изображений в память компьютера осуществляется с помощью видеодатчиков. Видеодатчик переводит оптическое распределение яркости изображения в электрические сигналы и далее в цифровые коды. Поскольку изображение является функцией двух пространственных переменных, а электрический сигнал является функцией одной переменной - времени, то для преобразования используется развертка. Например, при использовании телевизионной камеры изображение считывается по строкам: строка за строкой. При этом в пределах каждой строки зависимость яркости от пространственной координаты x преобразуется в пропорциональную зависимость амплитуды электрического сигнала от времени t. Переход от конца предыдущей строки к началу следующей осуществляется практически мгновенно. Широкое применение в качестве видеодатчиков находят также матрицы фотодиодов и матрицы приборов с зарядовой связью. При использовании матричных видеодатчиков изображение как бы наблюдается сквозь экран с множеством прозрачных ячеек. Число таких ячеек для современных видеодатчиков весьма велико и составляет величину 1024 i 1024 и более (см. рис. 5).

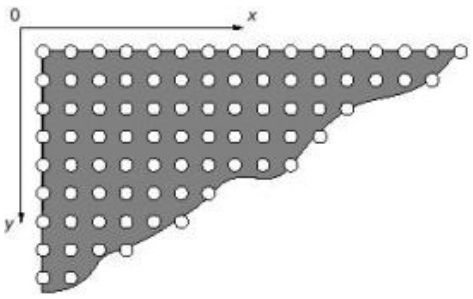



**Рис. 5.** Фрагмент матричного видеодатчика.

Исходное изображение, как уже отмечалось, представляет собой функцию двух непрерывных аргументов. В то же время цифровая память компьютера способна хранить только массивы данных. Поэтому ввод изображения в компьютер неизбежно связан с дискретизацией изображений по пространственным координатам и по яркости.

## 2. ДИСКРЕТНЫЕ ПРЕДСТАВЛЕНИЯ ИЗОБРАЖЕНИЙ

### 2.1. Дискретизация изображений

Рассмотрим непрерывное изображение f (x, y) - функцию двух пространственных переменных x и y на ограниченной прямоугольной области (рис. 6).

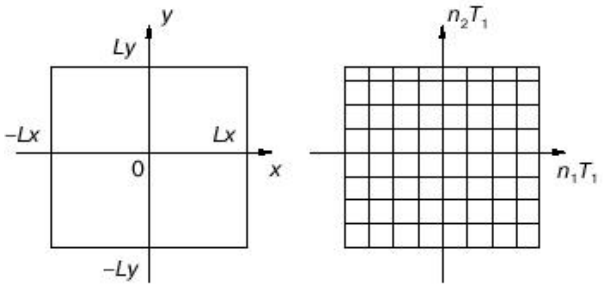



**Рис. 6.** Переход от непрерывного изображения к дискретному.

Введем понятие шага дискретизации T1 по пространственной переменной x и T2 по переменной y. Например, можно представить, что в точках, удаленных друг от друга на расстояние T1 по оси x, расположены точечные видеодатчики. Если такие видеодатчики установить по всей прямоугольной области, то изображение окажется заданным на двумерной решетке:

Для сокращения записи обозначим

f (n1T1 , n2T2) Ï f (n1 , n2).

Функция f(n1 , n2) является функцией двух дискретных переменных и называется двумерной последовательностью. То есть дискретизация изображения по пространственным переменным переводит его в таблицу выборочных значений. Размерность таблицы (число строк и столбцов) определяется геометрическими размерами исходной прямоугольной области и выбором шага дискретизации по формуле

где [ " ] обозначает целую часть числа.

Если область определения непрерывного изображения - квадрат Lx = Ly = L и шаг дискретизации выбран одинаковым по осям x и y (T1 = T2 = T ), то

Mx = My = M

и размерность таблицы составляет M 2.

Элемент таблицы, полученной путем дискретизации изображения, называют пиксел. Рассмотрим пиксел f (n1 , n2). Это число принимает непрерывные значения.

Память компьютера способна хранить только дискретные числа. Поэтому для записи в памяти непрерывная величина f должна быть подвергнута аналогово-цифровому преобразованию с шагом D (см. рис. 7).

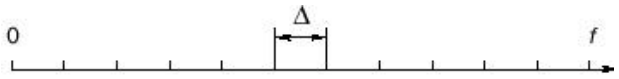



**Рис. 7.** Квантование непрерывной величины.

Операцию дискретизации непрерывной величины по уровням часто называют квантованием. Число уровней квантования равно

В практических задачах обработки изображений величина K варьируется в широких пределах от K = 2 ("бинарные" (черно-белые) изображения) до K = 210 и более (практически непрерывные значения яркости). Наиболее часто выбираются K = 28, при этом пиксел изображения кодируется одним байтом информации. Из всего вышеуказанного делаем вывод, что пикселы, хранящиеся в памяти компьютера, представляют собой результат дискретизации исходного непрерывного изображения по аргументам и по уровням. Ясно, что шаги дискретизации T1 ,T2 и D должны выбираться достаточно малыми, для того чтобы погрешность дискретизации была незначительна и цифровое представление сохраняло основную информацию об изображении.

При этом следует помнить, что чем меньше шаг дискретизации и квантования, тем больший объем данных об изображении должен быть записан в память компьютера. Рассмотрим в качестве иллюстрации этого утверждения изображение на слайде размером 50 i 50 мм, которое вводится в память с помощью цифрового измерителя оптической плотности (микроденситометра). Если при вводе линейное разрешение микроденситометра (шаг дискретизации по пространственным переменным) составляет 100 мкм, то в память записывается двумерный массив пикселов размерности M 2 = 500 i 500 = 25 i 104. Если же шаг уменьшить до 25 мкм, то размеры массива возрастут в 16 раз и составят M 2 = 2000 i 2000 = = 4 i 106. Используя квантование по 256 уровням, то есть кодируя найденный пиксел

байтом, получаем, что в первом случае для записи необходим объем 0,25 мегабайт памяти, а во втором случае - 4 мегабайта.

С физической точки зрения выбор шага дискретизации диктуется шириной пространственного спектра изображения. Чем больше ширина спектра W , тем меньше шаг дискретизации T. Практически при дискретизации стремятся удовлетворить соотношению

**2.2. 2D-последовательности**

Рассмотрим несколько практически важных 2D-последовательностей, имеющих аналитическое выражение.

1) Цифровой единичный импульс

Нетрудно заметить, что эта последовательность подобна дельта-функции (8). Произвольная последовательность F(n1 , n2) может быть представлена в виде

(сравним с формулой (12)).

2) Цифровой единичный скачок

- функция, которая принимает единичные значения в правом верхнем квадранте координатной плоскости и нулевое значение в других квадрантах.

3) Экспоненциальная последовательность

4) Комплексная экспонента

exp(n1 , n2) = exp[i(w1n1 + w2n2)],

где w1 , w2 имеют смысл пространственных частот.

**2.3. 2D-системы**

С математической точки зрения, 2D-система - это правило, которое ставит в соответствие 2D-входной последовательности f (n1 , n2) 2D-выходную последовательность g(n1 , n2).

Напомним, что мы рассматриваем линейные пространственно-инвариантные системы. Подавая на вход системы функцию u0(n1 , n2), на выходе получаем функцию h(n1 , n2), которая называется импульсной реакцией системы.

Импульсная реакция позволяет записать связь между входной и выходной двумерными последовательностями системы в виде

(сравним с (16)).

Формула 2D-свертки имеет большую вычислительную сложность. Для иллюстрации рассмотрим

**ЛИТЕРАТУРА**

1. Прэтт У. Цифровая обработка изображений. В двух книгах. М.: Мир, 1982.

# КОМПЬЮТЕРНАЯ ОБРАБОТКА ИЗОБРАЖЕНИЙ.

ЧАСТЬ 2. МЕТОДЫ И АЛГОРИТМЫ

В.А. Сойфер

Самарский государственный аэрокосмический университет

1996

Исходный URL: http://www.pereplet.ru/obrazovanie/stsoros/49.html

## 3. ПОЭЛЕМЕНТНЫЕ ПРЕОБРАЗОВАНИЯ ИЗОБРАЖЕНИЙ

### 3.1. Общее описание метода

При цифровой обработке изображения обычно используется его представление в памяти в виде матрицы пикселов f(m1 , m2), 0 < m1 < M1 - 1, 0 < m2 < M2 - 1. Обработка изображения в общем случае заключается в выполнении какого-либо преобразования указанной матрицы, в результате которого формируется набор ее числовых характеристик или новое, обработанное изображение - g(n1 , n2), 0 < n1 < N1 - 1, 0 < n2 < N2 - 1. Преобразование может касаться значений элементов или их координат (индексов), выполняться над матрицей в целом, группой элементов или над каждым элементом в отдельности.

В данном разделе рассматривается простейший вид цифровой обработки изображений, заключающийся в выполнении одного и того же функционального преобразования для каждого элемента матрицы вне зависимости от его положения и значений других (соседних) элементов. Такая обработка получила название поэлементного преобразования изображений. Она переводит значение каждого элемента f в новое значение g в соответствии с заданной функциональной зависимостью

g = g(f ).

Размеры входного и выходного изображения здесь, очевидно, совпадают (M1 = N1 , M2 = N2). При практической реализации поэлементных преобразований можно непосредственно вычислять каждое значение преобразованного элемента в соответствии с конкретным видом функции (30). Однако для достаточно сложных функций такое построение процедуры обработки оказывается неудобным из-за больших затрат машинного времени на вычисления. Скорость обработки возрастает при переходе к табличному заданию функции преобразования. Алгоритм работы с таблицей очень прост: по значению f вычисляется адрес (номер строки) таблицы с выходным значением g. Очевидные преимущества такого подхода: высокое быстродействие, а также гибкость процедуры обработки (таблица преобразования по сути является параметром процедуры и может легко меняться); недостаток: приближенность результатов из-за ограниченного числа строк таблицы. Несмотря на простоту, метод поэлементных преобразований позволяет решить довольно много прикладных задач улучшения качества и анализа изображений. Рассмотрим некоторые из них.

### 3.2. Линейное контрастирование

Изображения, вводимые в компьютер, часто являются малоконтрастными, то есть у них вариации функции яркости малы по сравнению с ее средним значением. Реальный динамический диапазон яркостей [fmin , fmax] для таких изображений оказывается намного меньше допустимого диапазона (шкалы яркости). Задача контрастирования заключается в "растягивании" реального динамического диапазона на всю шкалу. Контрастирование можно осуществить при помощи линейного поэлементного преобразования

g = af + b.

Параметры этого преобразования a, b нетрудно определить, исходя из требуемого изменения динамического диапазона. Если в результате обработки нужно получить шкалу [gmin , gmax], то, как следует из (31),

gmin = afmin = b,

gmax = afmax = b.

Отсюда

При диалоговой обработке изображений иногда проще не определять параметры преобразования (31), а непосредственно строить его в табличной форме, ориентируясь на границы распределения вероятностей функции яркости.

### 3.3. Пороговая обработка

Некоторые задачи обработки изображения связаны с преобразованием полутонового изображения (то есть такого, которое имеет много градаций яркости) в бинарное (двухградационное). Такое преобразование осуществляется в первую очередь для того, чтобы сократить информационную избыточность изображения, оставить в нем только ту информацию, которая нужна для решения конкретной задачи. В бинарном изображении должны быть сохранены интересующие нас детали (например, очертания изображенных объектов) и исключены несущественные особенности (фон).

Пороговая обработка полутонового изображения заключается в разделении всех элементов изображения на два класса по признаку яркости, то есть в выполнении поэлементного преобразования вида

где f0 - некоторое "пороговое" значение яркости.

При выполнении пороговой обработки основной вопрос состоит в выборе порога f0 . Пусть полутоновое изображение содержит интересующие нас объекты одной яркости на фоне другой яркости (типичные примеры: машинописный текст, чертежи, медицинские пробы под микроскопом и т. д.). Тогда в идеале плотность распределения яркостей должна выглядеть как две дельта-функции (рис. 8а). В данном случае задача установления порога тривиальна: в качестве f0 можно взять любое значение между "пиками". На практике, однако, встречаются определенные трудности, связанные с тем, что, во-первых, изображение искажено шумом и, во-вторых, как для объектов, так и для фона характерен некоторый разброс яркостей. В результате пики функции плотности распределения "расплываются", хотя обычно ее бимодальность сохраняется (рис. 8б). В такой ситуации можно выбрать порог f0 , соответствующий положению минимума между модами, то есть использовать функцию поэлементного преобразования, показанную на рис. 8в.

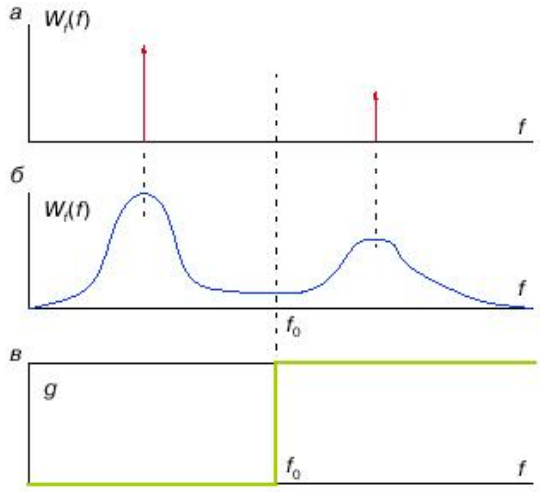



**Рис. 8.** Совместное распределение яркости двух объектов.

### 3.4. Препарирование

Широкий класс процедур обработки изображений заключается в их препарировании, то есть в приведении к такому виду, который, возможно, весьма далек от естественного, но удобен для визуальной интерпретации или дальнейшего машинного анализа.

Многие операции препарирования могут осуществляться при помощи поэлементных преобразований специальных видов. Так, частным случаем препарирования является пороговая обработка, рассмотренная выше. Перечислим некоторые другие используемые преобразования.

Очевидным обобщением пороговой обработки является преобразование яркостного среза (рис. 9а). Оно позволяет выделить определенный интервал диапазона яркостей входного изображения. Перемещая "рабочий" интервал по шкале и меняя его ширину, можно произвести визуальный анализ отдельных изображенных объектов, различающихся по яркости. Детали, не попадающие в указанный интервал, то есть относящиеся к "фону", будут подавлены. На рис. 9б

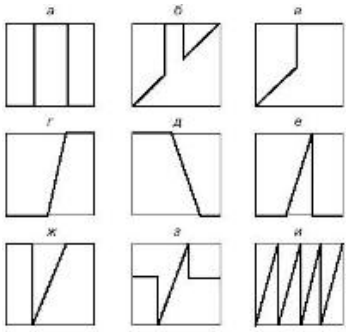


Рис. 9. Преобразование яркостного среза.

приведен вариант яркостного среза с сохранением фона.
В данном случае изображение в целом сохраняется, но на нем "высвечиваются" участки, попавшие в заданный интервал яркостей. Если этот интервал примыкает к границе шкалы яркости, то получаем преобразование так называемой неполной пороговой обработки (рис. 9в).

Контрастное масштабирование в своем простейшем варианте совпадает по смыслу с линейным контрастированием, рассмотренным в п. 3.2, здесь "рабочий" интервал яркостей растягивается на весь диапазон допустимых значений (рис. 9г). В других случаях контрастное масштабирование может быть связано с обращением функции яркости, то есть получением "негатива" (рис. 9д), представлением "рабочего" интервала яркостей на однородном фоне: черном (рис. 9е), белом (рис. 9ж) или сером (рис. 9з) и т. д.

Пилообразное контрастное масштабирование иллюстрирует рис. 9и. Как показывает практика, если изображение состоит из нескольких крупных областей с медленно меняющимися (по плоскости) значениями яркости, то такое преобразование почти не разрушает целостности его восприятия и в то же время резко увеличивает контрастность плохо различимых мелких деталей.

## 4. ВЫДЕЛЕНИЕ КОНТУРОВ

### 4.1. Постановка задачи выделения контуров

Исследованиями психологов установлено, что с точки зрения распознавания и анализа объектов на изображении наиболее информативными являются не значения яркостей объектов, а характеристики их границ - контуров. Другими словами, основная информация заключена не в яркости отдельных областей, а в их очертаниях. Задача выделения контуров состоит в построении изображения именно границ объектов и очертаний однородных областей.

На рис. 10а, б показаны соответственно исходное изображение, состоящее из областей различной яркости, и его графический вариант, состоящий только из границ этих областей.

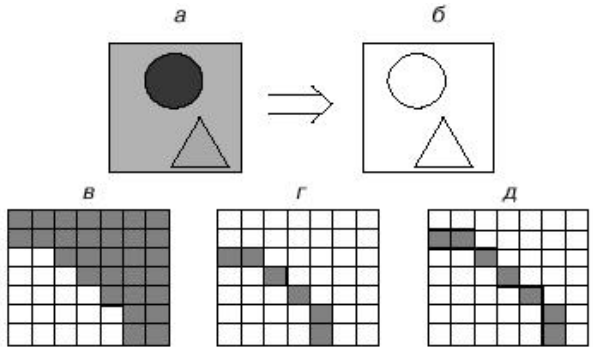



**Рис. 10.** К определению понятия контура.

Будем называть контуром изображения совокупность его пикселов, в окрестности которых наблюдается скачкообразное изменение функции яркости. Так как при цифровой обработке изображение представлено как функция целочисленных аргументов, то контуры представляются линиями шириной, как минимум, в один пиксел. При этом может возникнуть неоднозначность в определении линии контура, как это показано на рис. 10г, д, для исходного изображения с перепадом яркости (рис. 10в).

Если исходное изображение, кроме областей постоянной яркости, содержит участки с плавно меняющейся яркостью, то введенное определение контура остается справедливым, однако при этом не гарантируется непрерывность контурных линий: разрывы контуров будут наблюдаться в тех местах, где изменение функции яркости не является достаточно резким (этот эффект иллюстрируется на рис. 11).

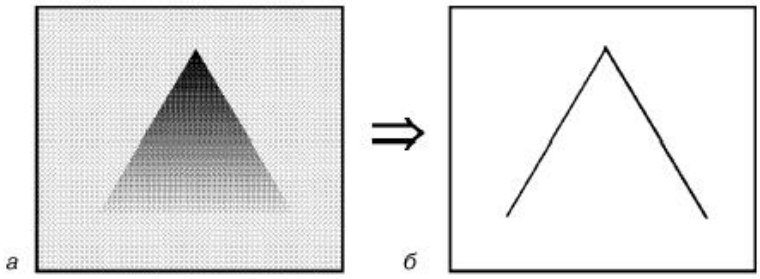



**Рис. 11.** Разрыв контура.

С другой стороны, если на "кусочно-постоянном" изображении присутствует шум, то, возможно, будут обнаружены "лишние" контуры в точках, которые не являются границами областей.

При разработке алгоритмов выделения контуров нужно учитывать указанные особенности поведения контурных линий. Специальная дополнительная обработка выделенных контуров позволяет устранять разрывы и подавлять ложные контурные линии.

Общую процедуру построения бинарного изображения границ объектов иллюстрирует блок-схема, представленная на рис. 12.

Исходное изображение f подвергается линейной или нелинейной обработке, с тем чтобы выделить перепады яркости. В результате этой операции формируется изображение e, функция яркости которого существенно отличается от нуля только в областях резких изменений яркости изображения f. Затем в результате пороговой обработки из изображения e формируется графический (контурный) препарат g. Правильный выбор порога на втором этапе должен производиться из следующих соображений. При слишком высоком пороге могут появиться разрывы контуров, а слабые перепады яркости не будут обнаружены. При слишком низком пороге из-за шумов и неоднородности областей могут появиться ложные контуры. Других особенностей пороговая обработка не имеет. Поэтому обратим основное внимание на первую операцию - выделение перепадов яркости (контуров) - и рассмотрим основные методы выполнения этой операции.

**4.2. Градиентный метод**

Одним из наиболее простых способов выделения границ является пространственное дифференцирование функции яркости. Для одномерной непрерывной функции яркости f(x) этот способ иллюстрирует рис. 13.

Для двумерной функции яркости f(x, y)перепады в направлениях x и y регистрируются частными производными ?f(x, y)/ ?x и ?f(x, y)/ ?y , которые пропорциональны скоростям изменения яркости в соответствующих направлениях. Выделение перепадов яркости в двумерном случае иллюстрирует рис. 14, на нем однократная штриховка соответствует нулевому значению функции, двухкратная - отрицательному значению, отсутствие штриховки - положительному). Видим, что подчеркивание контуров, перпендикулярных к оси x, обеспечивает производная ?f(x, y)/ ?x (рис. 14б), а подчеркивание контуров, перпендикулярных к оси y, - ?f(x, y)/ ?y (рис. 14в).

В практических задачах требуется выделить контуры, направление которых является произвольным. Для этих целей можно использовать модуль градиента функции яркости

который пропорционален максимальной (по направлению) скорости изменения функции яркости в данной точке и не зависит от направления контура.

Модуль градиента в отличие от частных производных принимает только неотрицательные значения, поэтому на получающемся изображении (рис. 14г) точки, соответствующие контурам, имеют повышенный уровень яркости.

Для цифровых изображений аналогами частных производных и модуля градиента являются функции, содержащие дискретные разности, например:

Таким образом, операция выделения контуров заключается в выполнении нелинейной локальной обработки изображений "окном" 2 i 2 (без одной точки):

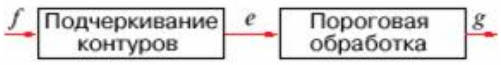



**Рис. 12.** Процедура выделения контуров.

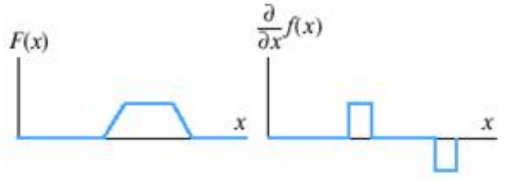



**Рис. 13.** Дифференцирование функции яркости.

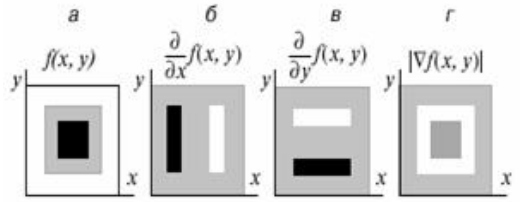



**Рис. 14.** Выделение перепадов яркости.

## 5. ОБРАБОТКА ИЗОБРАЖЕНИЙ СКОЛЬЗЯЩИМ ОКНОМ

### 5.1. Постановка задачи

Рассмотрим поле изображения, которое сканируется скользящим "окном", покрывающим одновременно несколько пикселов. При этом в окне оказывается небольшой фрагмент изображения. При перемещении окна фрагмент меняется. Постепенно окно сканирует все поле изображения. Сканирование изображения скользящим окном иллюстрируется рис. 15 (изображение представляет собой полукольцо). На рис. 15 показано три различных положения окна.

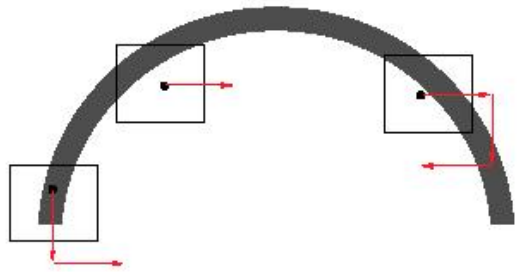
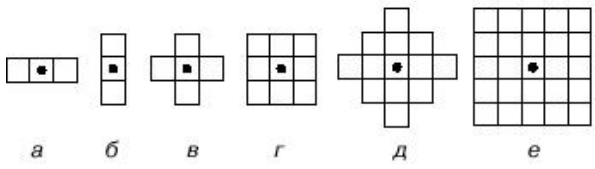


**Рис. 15.** Сканирование изображения скользящим окном.

Все пикселы, попадающие в окно, обрабатываются по некоторым правилам. Результатом обработки является пиксел выходного изображения, обычно соответствующий центру окна.




Наиболее распространенные виды окон представлены на рис. 16.

**Рис. 16.** Распространенные виды окон.

Все пикселы, попадающие в окно, обрабатываются по некоторым правилам. Результатом обработки является пиксел выходного изображения, он обычно соответствует центру окна:

Из (37) следует, что поэлементное преобразование является частным случаем локальной обработки изображений, соответствующей размерам окна 1 i 1. В общем случае функция (37) является нелинейной и может иметь весьма сложную структуру. Ясно, что для ее реализации нельзя использовать табличный метод (как при поэлементной обработке). Для большого числа практических задач обработку можно представить в виде наложения линейной обработки окном и поэлементного преобразования. Указанная процедура имеет вид

где e(n1 , n2) - промежуточное изображение, a(k1 , k2) - коэффициенты линейной маски; c - коэффициент постоянного смещения.

Соотношение (38) описывает двумерную свертку функции яркости исходного изображения f(n1 , n2) с импульсной реакцией a(k1 , k2) некоторой двумерной системы. Выбирая определенным образом коэффициенты маски a(k1 , k2), а также ее размеры, можно выполнить различные операции по улучшению качества изображения: повышение резкости, сглаживание шумов, подчеркивание контуров, низкочастотную или высокочастотную фильтрацию и т. д. Из соображений простоты обычно применяются окна малых размеров 3 i 3 или 5 i 5.

Коэффициент c в (38) не имеет принципиального значения для выполняемых операций. Он необходим для "компенсации" получающихся отрицательных значений отсчетов яркости.

При обработке изображений с использованием линейной маски следует учитывать "краевые эффекты". С практической точки зрения исходное и обработанное изображения удобно считать имеющими одинаковые размеры M1 i M2 . Однако из (38) следует, что в этом случае получающееся изображение не определено вблизи краев, так как окно выходит за границу изображения при некоторых значениях индексов m и n. В этом случае обычно считают, что неопределенные значения элементов яркости вне поля изображения (n1 < 0, или n2 < 0, или n1 > M1 , или n2 > M2) равны константе (например, нулю) или значениям ближайших определенных элементов.

**5.2. Модели помех при регистрации изображений**

Никакая система регистрации не обеспечивает идеального качества изображений исследуемых объектов. Изображения в процессе формирования их изображающими системами (фотографическими, голографическими, телевизионными) обычно подвергаются воздействию различных случайных помех или шумов. В отличие от так называемых детерминированных искажений, которые часто описываются поэлементными функциональными преобразованиями исходного изображения, для описания случайных воздействий используют модели аддитивного, импульсного и мультипликативного шумов.

Наиболее распространенным видом помех является случайный аддитивный шум, статистически независимый от видеосигнала. Модель аддитивного шума используется тогда, когда сигнал на выходе изображающей системы или на каком-нибудь промежуточном этапе преобразования может рассматриваться как сумма полезного сигнала и некоторого случайного сигнала (шума). Модель аддитивного шума хорошо описывает действие зернистости фотопленки, флуктуационный шум в радиотехнических системах, шум квантования в аналогоцифровых преобразователях и т.п.

Если действие шума сказывается не по всей протяженности поля изображения, а только в случайно расположенных точках, в которых значения функции яркости заменяются случайными величинами, то шум называют импульсным. На изображении такие помехи выглядят изолированными контрастными точками. Будем считать, что искаженные точки равномерно распределены по всему полю изображения, а яркость искаженных точек имеет равномерное распределение в некотором диапазоне. Импульсный шум характерен для систем передачи изображений по радиоканалам с использованием нелинейных методов модуляции, а также для цифровых систем передачи и хранения изображений. В частности, импульсный шум присущ устройствам ввода изображений с телевизионной камеры.

Более общей линейной моделью наблюдения изображения в условиях помех является модель, учитывающая наряду с аддитивным шумом динамические пространственные искажения. Если такие искажения можно описать пространственно-однородной (инвариантной к сдвигу) линейной системой с импульсной характеристикой h(k1 , k2), то модель наблюдения принимает вид

где u(n1 , n2) - случайный шум.

Модель (39) описывает искажения, вызванные движением системы регистрации относительно объекта, турбулентностью атмосферы, аберрациями оптической системы, неточностью фокусировки и т.п.

5.3. Алгоритмы линейной фильтрации изображений

Рассмотрим схему искажения и фильтрации (восстановления) изображений, представленную на рис. 17.


**Рис. 17.** Модель искажения и восстановления изображений.

Целью восстановления искаженного изображения y(n1 ,n2) является получение из него при помощи некоторой обработки изображения , которое близко к идеальному изображению x(n1 ,n2) по заданному критерию. Получающееся в результате обработки изображение будем называть оценкой исходного (идеального) изображения x(n1 ,n2). Определим ошибку оценивания в каждой точке изображения:

а также среднюю квадратичную ошибку (СКО) через ее квадрат, то есть дисперсию ошибки:

Критерий минимума квадрата СКО (e- 2 min) является наиболее универсальным и распространенным критерием качества восстановления при проектировании алгоритмов фильтрации изображений из-за математической простоты. Однако этот критерий имеет недостаток, заключающийся в том, что он не всегда согласуется с субъективным (психовизуальным) критерием качества, основанным в основном на точности передачи контуров.

Указанный критерий является конструктивным и позволяет теоретически рассчитывать оптимальные (дающие минимумы квадрата СКО) алгоритмы фильтрации при рассмотренных моделях наблюдения. Однако оптимальные алгоритмы оказываются весьма сложными для расчета и реализации. В автоматизированных системах обработки изображений предпочтение отдается так называемым квазиоптимальным алгоритмам, которые дают минимум квадрата СКО в некотором классе алгоритмов с заданной структурой и незначительно отличаются от оптимальных по этому критерию. Обычно спектр шума содержит более высокие пространственные частоты, чем спектр идеального изображения. Этот факт наводит на мысль, что простая низкочастотная фильтрация может служить эффективным средством подавления шумов. В принципе любой фильтр с неотрицательными коэффициентами обладает сглаживающими свойствами. Можно предложить следующие сглаживающие маски:

Коэффициенты масок нормированы:

с тем чтобы процедура подавления помех не вызывала смещения яркости исходного изображения. Маски (42) отличаются степенью сглаживания шумов (у маски A1 она максимальная, у A3 - минимальная). Выбор коэффициентов маски должен производиться экспериментально. При увеличении степени сглаживания шумов происходит также подавление высокочастотной составляющей полезного изображения, что вызывает исчезновение мелких деталей и размазывание контуров. Если требуемая степень сглаживания с применением маски размера 3 i 3 не достигается, то следует использовать сглаживающие маски больших размеров (5 i 5, 7 i 7)....

## ЗАКЛЮЧЕНИЕ

Компьютерная обработка изображений как фундаментальное научное направление является неисчерпаемой. Это направление опирается на математику, физику, биологию, информатику. Методы и средства компьютерной обработки изображений имеют самые разнообразные применения: наука, техника, медицина, социальная сфера. Практически уже сейчас прогресс общества, особенно в сфере здравоохранения, во многом зависит от достижений компьютерной обработки изображений. В дальнейшем роль компьютерной обработки изображений в жизни человека будет возрастать еще больше.

## ЛИТЕРАТУРА

1. Прэтт У. Цифровая обработка изображений. В двух книгах. М.: Мир, 1982.