Парадигма развития науки

А. Е. Кононюк

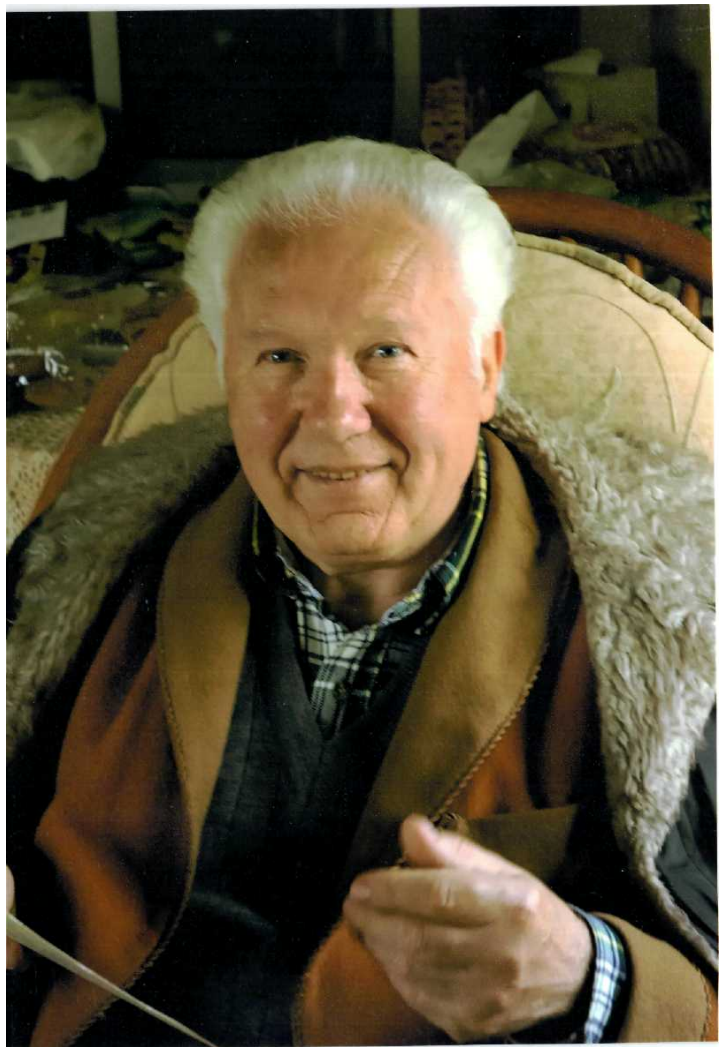
Основы фундаментальной теории искусственного интеллекта

Книга 3

Зрительное восприятие изображений искусственным интеллектом

Часть 1

Киев
«Освіта України»
2017



Кононюк Анатолий Ефимович

Структурная схема
парадигмы развития науки

УДК 51 (075.8)
ББК В161.я7
К65

Рецензент:
Н.К.Печурин - д-р техн. наук, проф. (Национальный авиационный университет).

Кононюк А. Е.
К213 Основы фундаментальной теория искусственного интеллекта. — В 20-и кн. Кн.3, ч.1. — К.:Освіта України. 2017.—608 с.

ISBN 978-966-373-693-8 (многотомное издание)
ISBN 978-966-373-694-7 (книга 3, ч.1)

Многотомная работа посвящена систематическому изложению общих формализмов, математических моделей и алгоритмических методов, которые могут быть исспользуемых при моделировании и исследованиях математических моделей объектов искусственного интеллекта.

Развиваются представления и методы решения, основанные на теориях эвристического поиска и автоматическом доказательстве теорем, а также процедуральные методы, базирующиеся на классе проблемно-ориентированных языков, сочетающих свойства языков программирования и автоматических решателей задач отображения искусственного интеллекта различными математическими средствами.

В работе излагаются основы теории отображения искусственного интеллекта такими математическими средствами как: множества, отношения, поверхности, пространства, алгебраические системы, матрицы, графы, математическая логика и др.

Для бакалавров, специалистов, магистров, аспирантов, докторантов всех специальностей.

УДК 51 (075.8)
ББК В161.я7

ISBN 978-966-373-693-8 (многотомное издание)
ISBN 978-966-373-694-7 (книга 3,ч. 1)

Оглавление

Введение

Любой объект является единственным в своем роде, но жизнь была бы невозможна, если бы мы относились к этой мысли слишком серьезно. Для многих целей мы находим удобным считать уникальные существа типичными: типичным персоналом станций обслуживания, типичными лифтерами или сборщиками мусора. Мы применяем знания характеристик общего класса для определения наших действий в конкретном случае, однако во избежание недоразумений следует научиться правильно проводить классификации. Именно эту способность Боурн, Брунер, Гуднау, Остин, Гилфорд, Хант предлагают считать основой разумного поведения. Для людей осуществление весьма тонких классификаций — такое обыденное дело, что мы не осознаем, насколько важен этот процесс. Мужчина, вернувшись домой, сразу узнает свою жену, ребенка, собаку. Очевидно, различия между тем, как жена и тем более собака выглядели утром и вечером, незначительны, но именно в них суть дела. Как мы определим для ЭВМ, какое изменение образа несущественно, а какое потребует новой классификации?

В своей работе Селфридж предложил осуществлять распознавание образов вычислением взвешенной суммы ряда „рекомендованных" классификаций, каждая из которых основана на различных характеристиках распознаваемого объекта (**признаках**). Хотя индивидуальные рекомендации могут носить почти случайный характер, система в целом может быть достаточно точной. Развитие этой идеи ведет к *параллельному* методу распознавания образов. Можно считать, что каждый объект имеет простейшее описание, **представляемое вектором**, элементы которого служат аргументами для ряда функций, и значения этих функций уже сосчитаны. Они в свою очередь служат аргументами для некоторой решающей функции, которая определяет окончательную классификацию.

Это описание, не отражающее полностью идей Селфриджа, представляет **распознавание образов как задачу классификации векторов, или точек в n-мерном пространстве.** В монографии Себестиана, связывающей задачи распознавания образов с математической теорией решений, и обзоре Мейзела методов распознавания образов этому подходу была придана более явная

форма. Указанные аспекты распознавания образов тесно связаны с классическими статистическими методами многомерного анализа.
К проблеме распознавания образов можно подходить также, отталкиваясь от аналогии с биологическими процессами. В некоторых условиях способности животных к распознаванию образов превышают способности любой машины, которую только можно построить. Для простоты мы рассмотрим лишь человеческие способности. При классификации, основанной на непосредственном сенсорном опыте, т. е. при распознавании лиц или произнесенных слов, люди легко превосходят технические устройства. **В „несенсорных" ситуациях действия людей на столь эффективны.** Например, люди не могут соперничать с программами классификации образов, если правильный способ классификации включает логические комбинации абстрактных свойств, таких, как цвет, размер и форма. Проблему осложняет то обстоятельство, что в этих ситуациях не ясно, по какой причине люди действуют нестабильно (т. е. очень хорошо или очень плохо). Какой тип распознавания образов используется в рассуждениях по аналогии? Этот вопрос представляет большой интерес.
Поскольку распознавание образов должно быть функцией нейронов животного, можно искать ключ к биологическому распознаванию образов в свойствах самого нейрона. Для многих целей нейрон можно рассматривать как пороговый элемент. Это значит, что он либо дает на выходе некоторую постоянную величину, если сумма его входов достигает определенного значения, либо же остается пассивным. Мак-Каллок и Питтс доказали, что любую вычислимую функцию можно реализовать с помощью должным образом организованной сети *идеальных нейронов* — пороговых элементов, логические свойства которых с достаточным основанием можно приписать реальному нейрону. Проблема состоит в том, можно ли найти какой-то разумный принцип реорганизации сети, позволяющий случайно объединенной вначале группе идеальных нейронов самоорганизоваться в „вычислительное устройство", способное решать произвольную задачу распознавания образов (Допустим, что каждый объект, подлежащий классификации, полностью описывается вектором, состоящим не более чем из k двоичных цифр. Тогда существует не более 2^k различимых объектов. Нам нужна функция, которая, получив одно из возможных описаний, дает 1 тогда и только тогда, когда описанный объект принадлежит некоторому классу, и 0 в противном случае. Один из выводов работы Мак-Каллока и Питтса заключается в том, что такую функцию можно построить на сети идеальных нейронов.)

Такой принцип реорганизации явился бы теорией обучения, применимой на уровне отдельного нейрона. Интуитивно ясно, что такой принцип должен существовать, поскольку можно видеть, что животные действительно обучаются новым правилам классификации и неверно было бы полагать, что в них с рождения „запаяны" все классификации, которым можно обучиться ? (Более серьезной кажется мысль о том, что животные предрасположены к обучению определенным типам функций. Нервная система в целом, несомненно, не совсем случайна. В то же время генетический код не содержит достаточной информации для определения всех деталей нервной системы взрослого человека.)
Нейрологическая теория обучения, выдвинутая канадским психологом Хеббом, хотя и была рассчитана вначале на использование в качестве модели, предназначенной только для психологии, оказала большое влияние на искусственный интеллект. Ее модификация применялась при определении принципов системы распознавания образов, получившей название *персептрон* (Розенблатт). Персептрон, или, точнее, персептроны (ибо то, что описал Розенблатт, было скорее принципом построения программ, нежели единственной программой) существуют и в форме программ, и как специально сконструированные аналоговые вычислительные машины. Значительные усилия были направлены на анализ общего класса систем распознавания образов, которые представляют **персептроны.** Были развиты **понятия систем *линейного распознавания* образов и систем *пороговой логики.* Первый термин относится к методу объединения индивидуальных решений распознающих элементов, соответствующих различным характеристикам, а второй — к использованию устройств, вырабатывающих постоянный сигнал, если уровень их входных сигналов превышает некоторую фиксированную величину.** Была разработана содержательная математическая теория (Нильсон), вершиной которой является анализ круга задач, решаемых с применением **линейных пороговых логических систем** (Минский, Пейперт).
В работах с персептронами основное внимание уделяется установке весов, приписываемых фиксированному множеству детекторов признаков. Это совпадает с формулировкой задачи, данной Селфриджем. Альтернативный метод распознавания образов сводится к поиску „хороших" признаков, которые так четко проводят разделение между классами, что определение подходящего правила приписывания весов признакам не представляет труда.
Системы распознавания образов, упоминавшиеся до сих пор, представляют собой по меньшей мере аналоги биологического рас-

познавания образов. В биологии **термин „распознавание образов" неявно относят к классификации на сенсорном уровне.** Это проявляется в постоянном обращении к **зрительным примерам распознавания образов.** Психологи употребляют термин „обучение понятиям" по отношению к задаче, которая математически совпадает с задачей распознавания образов, но отличается от нее в психологическом аспекте. Эту разницу легко проиллюстрировать следующим примером. Вообразим, что написана программа, различающая изображения мужчин и женщин. Что в этом случае означает „показать изображение вычислительной машине"? Это значит, что полутоновые оттенки небольших участков изображения каким-то способом закодированы в числах и полученный вектор взят в качестве входа для программы распознавания образов. Очень простой пример приведен на рис. 1.

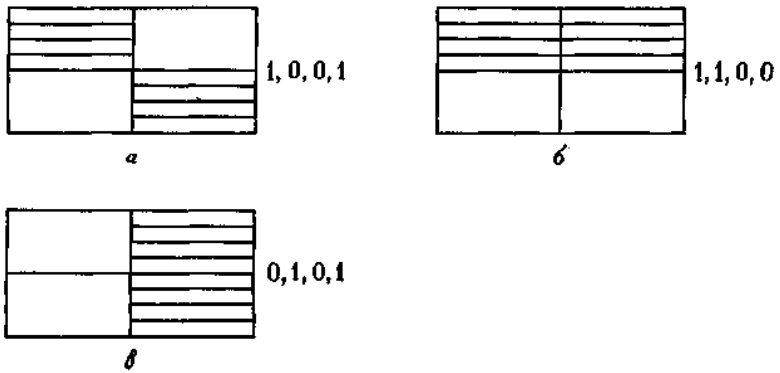


Рис. 1. Простой пример того, как изображение кодируется в вектор для задачи распознавания образов. Большой прямоугольник разбивается на четыре квадранта. Если верхний левый квадрант заштрихован, то первая компонента вектора равна 1, в противном случае — 0. Такие же правила применяются и к другим квадрантам и компонентам вектора.

Программа, далее, должна классифицировать уже векторы, а не изображения. Можно противопоставить ее программе, применяемой для классификации животных по их виду и роду. Обучение происходило бы путем сообщения программе описаний животных и правильного указания их вида и рода. На элементарном уровне **программа считывала бы множество признаков, описывающих животное, и сводила бы их к упорядоченному множеству кодов. Программа распознавания образов и в этом случае**

классифицировала бы векторы. Поскольку эти задачи эквивалентны на машинном уровне, видимо, было бы разумным, чтобы для вывода правил классификации образов в обоих случаях применялась одна и та же программа. Однако, если мы думаем о том, как человек мог бы осуществлятьобучение этим задачам, нам кажется, что **восприятие и познание могут как-то различаться.**

Те ученые, которые больше интересуются познавательной стороной задач распознавания образов, фактически разработали алгоритмы, заметно отличающиеся от полученных теми исследователями, кто при решении этой проблемы начинал с сенсорного уровня. Вместо проведения ряда параллельных опытов и оценивания общих результатов, **в алгоритмах логической классификации**, как правило, начинают с небольшого числа опытов и в зависимости от их результатов либо проводят второй тест, либо выполняют классификацию объекта. Характер второго теста может зависеть от результатов первого. Этот процесс, очевидно, можно распространить на любое число тестов. Правила классификации такого рода называются *последовательными процедурами принятия решений.* Эти методы еще недостаточно освещались в литературе по искусственному интеллекту, хотя такие же процедуры изучены подробно в статистике и исследовании операций. Следует отметить, что по крайней мере в одном случае, когда подход к логическому распознаванию образов был осуществлен с чисто статистических позиций (Морган и Сонквист), полученные алгоритмы вычисления были очень похожи на те, что появились после независимо выполненной работы Ханта, Марина и Стоуна в области искусственного интеллекта. В литературе по распознаванию образов все большее значение придается последовательным процедурам.

Очень может быть, что зрение человека — наиболее тонкая среди существующих систем распознавания образов. Очевидно, что вычислительная машина должна иметь возможность рассматривать зрительную сцену и анализировать ее так, как это делает человек. Проблема машинного зрения представляет собой ряд очень трудных задач. Анализ зрительных сцен оказывается в сущности невозможным, если анализирующее устройство не содержит логическую модель наблюдаемой сцены, позволяющую разрешать неоднозначности изображений на входе. Это заключение едва ли будет неожиданным для психологов, которые занимались константностью восприятия (например, константность цветного восприятия — это надежное восприятие человеком цвета объекта, не зависящее от внешнего освещения при более или менее нормальных условиях) и иллюзиями, и не удивительно, что некоторые предложения о том, как ЭВМ должна обрабатывать

зрительные данные, заимствованы из исследований по зрению человека. Точно так же стремление наделить вычислительную машину «глазами» вызвало идеи, которые могли бы использоваться при создании теорий человеческого зрения, хотя характер этого соответствия не совсем ясен. В настоящее время мы в состоянии дать лишь беглый обзор этой быстро меняющейся области, основные методы которой, возможно, еще не сформировались.

1. Общие соображения при распознавании объектов (ситуаций, сцен)

1.1. Классификация

Классификация лежит в основе интеллектуальной деятельности. Мы так легко классифицируем, что редко задумываемся над тем, насколько это существенно или насколько тонки производимые нами классификации. Вы в состоянии узнать свое имя, произнесенное мужчиной или женщиной, простуженным человеком или здоровым, на фоне шума самолетных двигателей или сонаты Бетховена. Если бы вам пришлось анализировать физические воздействия, которым в каждом случае подвергается ваше ухо, то вы не нашли бы в них почти никакого явного сходства. Теперь обратимся совершенно к другому случаю. Когда врач ставит диагноз, он не просто выбирает название для недуга больного. Он должен решить, принадлежит ли данный пациент классу пациентов, которому показано лечение X. И наконец, рассмотрим третью задачу классификации. Операторы радара или сонара должны решить, являются ли наблюдаемые на дисплее конфигурации результатом отражения от цели, действия фонового шума или отражения от неслучайной цели, но не той, за которой они следят. В общем случае нет совершенного правила, с помощью которого они могли бы это сделать, поскольку как цель, так и шум могут привести к почти любой конфигурации на дисплее. Различие состоит в вероятности, с которой цель или шум дадут определенные изображения.
С точки зрения психолога, эти примеры совершенно различны. **Первый обычно трактуется как задача восприятия, второй — как задача логического мышления и третий — как задача обнаружения сигнала.** Здесь и находится то место, где человек и вычислительная машина, возможно, отличаются друг от друга. Вполне может оказаться, что в каждом случае вовлекаются различные

психологические процессы. В то же время **эти задачи могут иметь одно и то же математическое описание, и они, возможно, должны анализироваться одинаковыми алгоритмами.** В каждом случае **наблюдателю дается набор *объектов,* которые можно описать заданием значений для каждого признака из известного множества *признаков***. Каждый объект принадлежит одному или более классам из некоторого фиксированного множества. **В задаче *классификации образов* наблюдатель должен применить установленное ранее правило, чтобы решить, какому классу принадлежит объект**. **В задаче *распознавания образов* это правило классификации должно вырабатываться на основе исследования множества объектов с известной принадлежностью различным классам.** Эти объекты в совокупности называются *обучающим множеством (organizing set*) или *выборкой.* В задаче *формирования образов* объекты предъявляются наблюдателю без указания их принадлежности классам. Наблюдатель должен самостоятельно построить соответствующее определение классов.
Далее мы будем в основном изучать задачу распознавания. (Внимательный читатель, возможно, уже заметил, что задача классификации эквивалентна задаче выяснения, является ли некоторая цепочка предложением в формальном языке.) В этой главе будет дана классификация задач распознавания образов. Приводимая схема классификации позволяет выделить отдельные подклассы задач распознавания.

1.1.1. Последовательная классификация образов

Рассматривается обучающаяся система последовательной классификации объектов, ориентированная на использование в системах ИИ. Поскольку в такого рода системах требуется осуществлять быструю реакцию на предъявленное изображение, возникает проблема ускорения процесса распознавания. Это значит, что требуется быстро находить значения признаков, характеризующих классифицируемые объекты, так как собственно процедура принятии решения по известным значениям признаков не требует много времени. Сокращение времени вычислений путем сокращения рзмерности пространства признаков является нежелательным, поскольку предлагаемая система распознавания является обучающейся, т.е. не предназначенной для решения какой-то одной фиксированной задачи распознавания. Могут встретиться различные объекты, поэтому признаки, которые могут пригодиться, нельзя игнорировать.

Для решения описанной проблемы предлагается проводить классификацию объектов в два шага. Схема такой процедуры представлено на рис. 2.

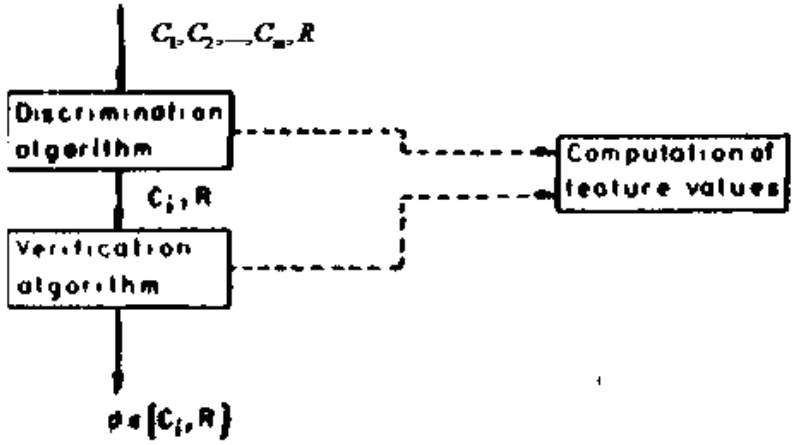


Рис. 2.

На первом шаге алгоритма распознавания проводится предварительная классификация поступающих образов. Поскольку на этой стадии используются лишь одномерные тесты для сравнения, проверка осуществляется весьма быстро. В результате работы классификатора после первого шага объект либо попадает в один из классов $C_1,.., C_m$, либо отвергается (класс отвергаемых объектов обозначен R).
На втором шаге проводится верификация решения, принятого на первом этапе. В результате верификации принятое на первом шаге решение либо признается окончательным, либо отвергается, и объект приписывается классу R . Алгоритм верификации основан на итеративном пересчете степени правильности решения μ в зависимости от значения очередного вычисленного признака. Параметр μ можно трактовать как отношение вероятности того, что данный объект принадлежит тому классу, к которому он был отнесен на первом шаге работы алгоритма, к вероятности того, что данный объект не принадлежит этому классу (то есть принадлежит классу R , поскольку пересмотр решения не допускается, и попыток отнести объект к другому классу не делается). Вначале параметру μ . приписывается начальное значение μ_0, которое не зависит от

используемых признаков. Последующие значения μ_i вычисляются по следующей формуле:

$$\mu_i = \mu_{i-1} \cdot c_{k,f(i)} \cdot p_k(x_{f(i)}) / p_R(x_{f(i)})$$

Здесь $f(i)$ означает i -ый признак, а $c_{k,f(i)}$ - множитель, зависящий от активированной в данный момент концевой вершины дерева решений, $p_k(x_{f(i)})$, $p_R(x_{f(i)})$- соответственно, функции плотности вероятности для k -го класса и для класса R, суженные на переменную $x_{f(i)}$.
Для ускорения вычислений предлагается аппроксимировать функции плотности вероятности простейшими распределениями, например треугольным. Кроме того, если в процессе вычислений μ_i достигла порогового значения, то вычисления можно прервать.
На стадии обучения требуется построить дерево решений для первого шага алгоритма распознавания и задать параметры для алгоритма верификации. Следует отметить, что обычные последовательные схемы обучения плохо подходят для практических применений, поскольку требуют слишком большого объема обучающей выборки. Вместо этого предлагается использовать следующую информацию: 1) априорную информацию об областях изменения значений признаков; 2) априорные вероятности встречи каждого из классов (включая класс R); 3) оценку стоимости вычислений данного множества признаков. В результате использования этой априорной информации можно значительно сократить объем обучающей выборки, ограничившись лишь немногими представителями каждого класса. Далее дерево решений строится путем разбиения пространства признаков на параллелепипеды, каждый из которых содержит объекты лишь одного класса. Дерево решений создается путем такого разделения пространства признаков, чтобы минимизировать среднюю стоимость вычислений.
Дерево решении конструируется рекуррентным образом, начиная с пустого дерева. На каждом шаге нетерминальная вершина либо замещается на концевую вершину (т.е. на вершину, в которой объекту присваивается номер класса), либо на такую нетерминальную (проверочную) вершину, из которой выходят две дуги к новым нетерминальным вершинам. В проверочной вершине выбирается для проверки такой признак и значения порогов, чтобы расщепить как можно меньше классов (параллелепипедов) и чтобы аппроксимированное значение средней величины дискриминации для поддерева, начинающегося с данной вершины, было бы минимальным. На рис. 3 и 4 приведено разбиение пространства признаков на параллелепипеды и соответствующее дерево решений.

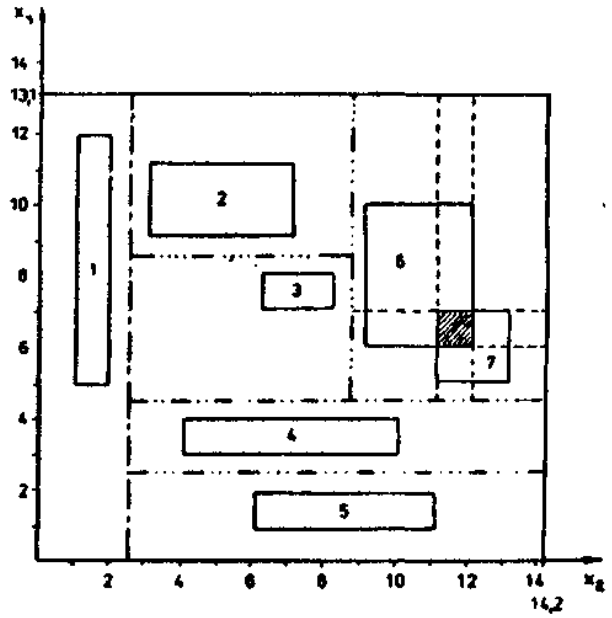


Рис. 3

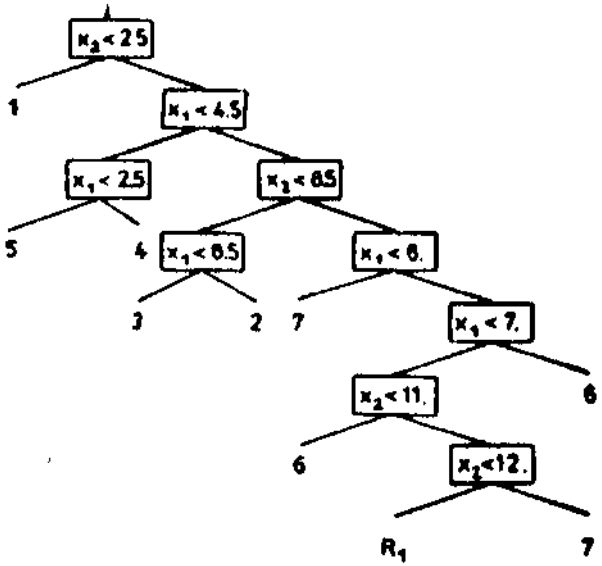


Рис. 4

Предложенная система классификации объектов реализована в зрительном анализаторе системы ИИ, предназначенном для распознавания плоских контурных изображений объектов.

1.1.2. Выбор оптимальной системы признаков в задачах распознавания изображений

При выполнении оперций сортировки объектов, подбора комплектов объектов, а также при сборочных операциях часто наблюдается ситуация, когда в поле зрения адаптивной СИИ поступают объекты различных типов, причем ЧИСЛО типов фиксировано. Система управления СИИ должна провести классификации объектов по типам путем выработки и анализа некоторых признаков. В качестве последних могут выступать такие характеристики, как площадь какой-либо проекции объекта или его параметр, отношение квадрата периметра к площади, число углов, спектральная, корреляционная или автокорреляционная функции и т.д.

Как правило , ни один из признаков не является достаточным для гарантированного выделения объектов каждого вида из ИХ совокупности. Поэтому классификация объектов возможна лишь при применении некоторой системы или набора признаков. Обычно технологический процесс накладывает определенные ограничения на систему управления СИИ. Это приводит к тому, что набор признаков должен быть выбран так, чтобы удовлетворять этим ограничениям, т.е. он должен *быть* оптимальным по одному или нескольких критериям. Наиболее часто самые жесткие требоввания налагаются на быстродействие систем управления СИИ, поэтому набор признаков будем выбирать по этому критерию.

Указанная задача может быть формализована и решена методом математического врогршшзрованм.

Пусть имеем *n* ТИПОВ объектов, а в качестве исходной системы выбирали некоторую избыточную совокупность из *m* признаков. Время выработки таких признаков для каждого типа объектов оценено заранее и приведено в табл.1.

Таблица 1

Признак j \ Тип объекта i	1	...	i	...	n
I	T_{11}	...	T_{1i}	...	T_{1n}
2	T_{21}	...	T_{2i}	...	T_{2n}
⋮	⋮		⋮		⋮
j	T_{j1}	...	T_{ji}	...	T_{jn}
⋮	⋮		⋮		⋮
m	T_{m1}	...	T_{mi}	...	T_{mn}

Построим матрицу различий объектов любых 2-х типов рассматриваемыми признаками. Элементами этой матрицы будут величины

$$a_{jI}=\begin{cases} \text{I, } \textit{если } j-\textit{й признак разделяет } \text{объекты } \textit{по типам} \\ i \textit{ их } \text{I}-\textit{й комбинации}; \\ 0, \textit{ в противном случае.} \end{cases}$$

Общее число попарных комбинаций будет

$$M = C_n^2 = \frac{n\,(n-1)}{2}.$$

При этом I =1 соответствует комбинация деталей I -го и 2-го типов и т.д. Указанная матрица приведена в табл.2.

Таблица 2

Признак j \ Комбинация I	1 (I,2)	... ...	n (I,n)	... ...	I (i_1,i_2)	... ...	M (n-1,n)
I	a_{11}	...	a_{1n}	...	a_{1I}	...	a_{1M}
2	a_{21}	...	a_{2n}	...	a_{2I}	...	a_{2M}
⋮	⋮		⋮		⋮		⋮
j	a_{j1}	...	a_{jn}	...	a_{jI}	...	a_{jM}
⋮	⋮		⋮		⋮		⋮
m	a_{m1}	...	a_{mn}	...	a_{mI}	...	a_{mM}

Пусть P_i - вероятность появления объекта i -го типа в поле зрения СИИ. Введем фиктивные переменные

$$X_j=\begin{cases} I, \text{если j-й признак был выбран при формировании} \\ \text{оптимального набора признаков;} \\ 0, \text{ в противном случае.} \end{cases}$$

Для сведения задачи формирования оптимального набора признаков к задаче математического программирования необходимо выбрать целевую функцию и систему ограничений.

Пусть необходимо минимизировать общее время выработки набора признаков при распознавании всей партии объектов при условии надежной их классификации. Тогда имеем целевую функцию

$$\sum_{i=1}^{n}\sum_{j=1}^{n} p_i T_{ji} x_j \longrightarrow min$$

и систему ограничений

$$\sum_{j=1}^{m} a_{jI} x_j \geq 1,\ I=1,\ldots,M.$$

Для удобства всегда можно выбрать единицы измерения T_{ij} , так, чтобы коэффициенты p_iT_{ji} можно было бы округлить до целых чисел. Тогда мы имеем задачу целочисленного линейного программирования с булевыми (0,1) переменными. Указанная задача может быть решена многими методами, например, с помощью аддитивного алгоритма

Балаша. В задачу могут быть введены дополнительные ограничения, например, ограничение на объем памяти и т.д. В результате решения получается оптимальный вектор с компонентами 0,1

$$X = \{x_1, x_2, \ldots, x_j, \ldots, x_m\},$$

показывающие, какие признаки должны быть включены в систему.

1.2. Характеризация задач распознавания образов

1.2.1. Общие положения

Разнообразие задач распознавания образов можно охарактеризовать тремя параметрами: способом, которым предъявляется наблюдателю обучающее множество, типом правила классификации образов, которое должен построить классификатор, и видом описания классифицируемых объектов. На рис. 5 показаны эти три параметра, объединенные в трехмерную схему.

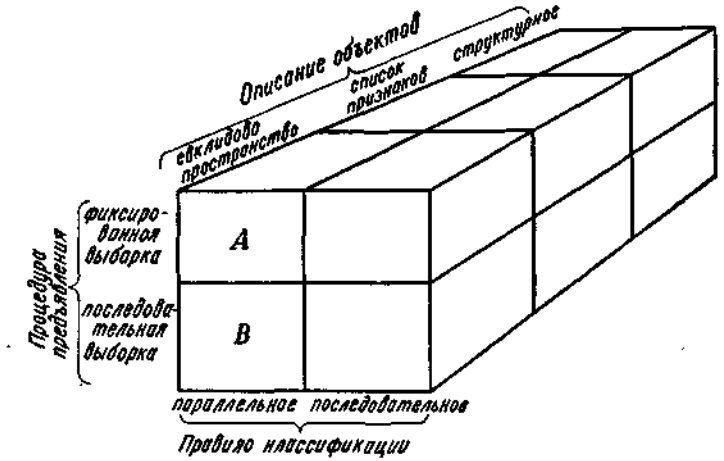


Рис. 5. Соотношение параметров в задачах распознавания образов.

Каждое ребро параллелепипеда представляет один из параметров. Каждая ячейка внутри параллелепипеда соответствует определенному классу задач. Например, ячейка, обозначенная на рисунке буквой *A,* включает в себя те задачи, в которых процедура классификации образов должна вырабатываться на основе информации, содержащейся в единственной выборке, при условии, что каждый из объектов можно

представить точкой в многомерном евклидовом *пространстве описаний,* и в предположении, что для построения правила классификации может потребоваться полное знание описания объекта. Мы можем противопоставить подобные задачи задачам из ячейки *B* (рис. 5), для которых евклидово пространство описаний и тип правила классификации сохраняются, а классификация по единственной выборке заменяется классификацией на основе последовательности выборок, при которой правило классификации уточняется после каждой выборки.

Три параметра, образующие параллелепипед, далее будут описаны подробнее, поскольку мы постоянно будем ссылаться на них при выработке методов решения определенных классов задач.

1.2.2. Предъявление обучающего множества

Рассматриваются два случая: распознавание образов, основанное на единственной выборке, и такие ситуации распознавания, когда используется последовательность выборок. В случае единственной выборки несколько объектов из известных классов предъявляются системе распознавания образов до начала классификации. На основе наблюдения этой выборки устройство распознавания вырабатывает правило классификации, применяемое затем к объектам, которые предопределяются указанной выборкой, но в ней не содержатся. Само правило классификации далее не меняется, даже когда наблюдаются ошибки классификации. В распознавании образов, использующем последовательность выборок, информация, получаемая первоначальной выборкой, является лишь предварительной, и она учитывается при построении соответствующего первоначального правила классификации. После выработки правила берется следующая выборка, к которой применяется имеющееся правило классификации. (Часто новая выборка состоит лишь из одного объекта.) Оценивается результат классификации и, если окажется нужным, отыскивается новое правило. Эту процедуру можно повторять сколько угодно, пока не будет удовлетворен некоторый критерий работы правила.

Выработку процедуры классификации на основе фиксированной выборки обычно относят к статистике, а не к искусственному интеллекту. Опубликовано много работ, в которых излагаются соответствующие методы, особенно для задач, в которых можно использовать евклидовы пространства описаний. Статистиков также интересовал и другой, тесно связанный с этим вопрос. Обнаруживают ли члены, относящиеся к разным группам, систематические различия

при измерении одной зависимой переменной? Этот вопрос очень важен для экспериментальных исследований, где „классификация" проводится с данными, получаемыми при меняющихся экспериментальных условиях.

Случай последовательной выборки представляет собой одну из наиболее активно изучающихся проблем искусственного интеллекта. Как правило, ее называют *машинным обучением.* Вероятно, название связано с тем, что непрерывное изменение правила классификации во многом аналогично способности большинства животных обучаться на опыте. Поскольку, **почти по определению, поступки животных разумны, система искусственного интеллекта должна обладать подобной способностью к обучению.** В самом деле, в ряде философских работ обучение на примерах выделяется как определение познавательной способности.

Если устройство распознавания образов может обучаться путем приспособления своих правил классификации к последовательным выборкам, возникает задача оценки полезности и стоимости каждого изменения. Обычно изменение правила классификации требует больше вычислений, чем классификация, использующая заданное правило. В самом деле, врачи не могут изменять свои диагностические процедуры после каждого больного, университеты не могут изменять свои правила приема каждый раз в зависимости от того, окончил данный студент университет или нет. С другой стороны, отсутствие возможности изменять ошибочное правило может привести к увеличению частоты ошибок классификации выше допустимых пределов. Вероятно, наиболее широко распространено в экспериментальных исследованиях или в распознавании образов изменение правила классификации всякий раз, когда происходит ошибка. Очевидная альтернатива — изменение правила классификации только в случае превышения частоты появления ошибок некоторого заранее установленного допустимого уровня. Такую процедуру можно считать лучшим приближением к распознаванию образов в практических ситуациях.

Можно построить устройство распознавания образов, в котором не учитывается информация об ошибках. Например, можно было бы рассматривать наблюдаемое среднее значений членов каждого класса по каждому параметру и классифицировать все новые объекты, оценивая, насколько близки они к текущей средней точке каждого класса. Интересно приложение данного метода, называемое „самостоятельным обучением", когда средние уточняются распознавателем образов в предположении, что классификации верны, и никак не учитывается обратная связь.

Способ, которым устройство распознавания образов изменяет пробное правило классификации, в большей мере определяется величиной информации, которую оно способно запомнить. В каждом случае, когда новый объект предъявляется для классификации, устройство распознавания получает какую-то информацию об окружающей среде — хотя бы только для улучшения оценки частоты появления объектов определенного типа. Как должна записываться такая информация? Существует два способа. Один из них — *детальный* метод, при котором информация о каждом объекте записывается в момент его предъявления. Например, врач обычно записывает результат каждого обследования и диагноз. Другой метод—*статистический;* он предполагает хранение обобщенной статистики, связанной с каждым классом и представляющей собой результат некоторого усреднения данных всех наблюдавшихся до настоящего момента случаев в соответствующем классе. Здесь примером может служить бюро погоды, строящее свои прогнозы на типичных показателях. Вместо хранения подробной информации об одном дне для обновления представлений о „типичном" дождливом или ясном дне используется информация о каждом новом дне.

Установление способа хранения информации и типа правила, которое нужно выработать, приводит к формулированию вычислительной проблемы: **определить алгоритм, эффективно отображающий множество возможных конфигураций „памяти" (т. е. множества всех возможных хранящихся записей о среде) в множество возможных правил классификации.** Для большинства случаев существуют несколько алгоритмов, которые могут отличаться друг от друга аспектами, весьма существенными при их практической реализации. Больше всего нас интересуют здесь такие свойства соответствующих процедур, как *сходимость, оптимальность* и *вычислительная сложность.*

Рассмотрим устройство распознавания образов, функционирующее так: вначале оно использует произвольно выбранное правило классификации. По мере получения информации путем демонстраций устройству объектов и указания классов, которым они принадлежат, оно вырабатывает последовательность новых правил классификации. **Если независимо от получаемой дополнительной информации устройство в некоторый момент перестает строить новые правила, говорят, что процедура *сходится* к окончательному правилу.** При описании алгоритма важны условия, гарантирующие сходимость. Нас может также интересовать, будут ли получаемые по пути к окончательному решению последовательные правила „больше похожи" (в некотором смысле) на окончательное решение, чем на предыдущие

правила в последовательности. **Устройство распознавания образов называют *оптимальным,* если гарантируется, что правило, к которому сходится процедура, минимизирует некоторую функцию, определяющую стоимость ошибочной классификации.** Часто эта функция есть просто число ошибок классификации, однако она может быть и более сложной. Иногда мы будем называть устройство распознавания образов *оптимальным в некотором классе устройств распознавания образов,* если оно вырабатывает окончательное правило, для которого значение функции ошибочной классификации не превышает значений функций ошибочной классификации, соответствующих окончательным правилам, вырабатываемым любым устройством из этого класса.

Понятие *вычислительной сложности* менее ясно и потому менее изучено. Интуитивно это понятие отражает трудность применения алгоритма. Последнюю можно было бы определить в терминах необходимой для выполнения рассматриваемого алгоритма машинной памяти, числа требуемых шагов вычислений, типом этих шагов или временем их выполнения. Существуют две важные разновидности понятия вычислительной сложности: сложность собственно алгоритма распознавания образов и сложность вырабатываемого им правила классификации. Сложность алгоритма распознавания образов тесно связана с вопросом сходимости; в обоих случаях нас интересует, насколько сложно достичь результата, применяя определенный метод. Вопрос о сложности правила классификации относится больше к практической приемлемости результатов, поскольку здесь для нас существенно, насколько хорошо полученное правило.

1.2.3. Правила классификации

Процедура распознавания образов — это алгоритм, формирующий правило классификации образов, исходя из обучающего множества. Очевидно, что тип используемого правила классификации будет определять **структуру процедуры распознавания**. Существуют два общих метода классификации: *параллельный* и *последовательный.* Для простоты предположим, что мы можем описать объект при помощи вектора символов. (В большинстве случаев это справедливо, хотя сущестуют исключения.) В параллельной процедуре производится ряд тестов над всеми компонентами вектора, а затем делается предположение о принадлежности объекта классу на основе объединенного результата этих тестов. В процедуре последовательной классификации сначала проверяется некоторое подмножество компонент вектора описания, а затем в зависимости от результатов

этих тестов или производится классификация, или выбираются новая совокупность тестов и новое подмножество компонент вектора описания, после чего указанный процесс повторяется.

Формальные выражения для параллельной и последовательной процедур достаточно прозрачны.

Пусть $\mathbf{x}=\{x_i\}$, $i=1, \ldots, n$,— вектор описания объекта, и объекты могут классифицироваться в c классов. В случае параллельной классификации существует множество $F=\{f_j\}$, $j=1, \ldots, c$, функций не более чем n переменных. В алгоритме классификации объект относится к классу j тогда и только тогда, когда

$$w_j = \max_k \{w_k\}, \quad k=1, \ldots, c, \qquad (1)$$

где

$$w_j = f_j(x_1, \ldots, x_n) \qquad (2)$$

Для j=1, ... , c. Термин „параллельный" здесь оправдан, поскольку не важно, в каком порядке вычисляются функции f_j, и, значит, если только для этого есть возможность, они могут вычисляться одновременно. Время, затрачиваемое на классификацию, будет определяться наибольшим временем, необходимым для вычисления любой из функций f_j, хотя общее количество вычислительных ресурсов, требуемых для проведения классификации, равно сумме ресурсов, требуемых для вычисления каждой функции f_j .

Селфридж привел наглядный пример параллельной процедуры. Предположим, что каждая функция f_j заменена маленьким демоном, задача которого — исследовать описание и выкрикивать название своего класса, если он считал, что объект относится именно к этому классу. Демон должен кричать громко, если он уверен в своем решении, и тихо, если не уверен. Однако общий шум, производимый демоном, будет зависеть не только от его стараний, но и от его способности кричать. Последнее определяет всемогущий (решающий) демон, который наделяет демонов первого порядка сильным или слабым голосом. После предъявления классифицируемого объекта каждый демон выкрикивает название своего класса с интенсивностью, зависящей от его собственных оценок и от силы данного ему голоса. Решающий демон, который ведет себя как председатель собрания, где проводится голосование, решает, название какого класса было выкрикнуто громче всех.

Процедуры последовательного решения несколько более громоздки, чтобы их можно было описать формально, но это можно сделать.

Удобнее представить эти процедуры в виде *дерева,* указывающего порядок, в котором должны производиться тесты.
(Деревом называется граф, в котором
(а) есть единственный узел, называемый *корнем,* не имеющий узлов над ним (т. е. у корня нет предков),
(б) все остальные узлы имеют точно по одному предку,
(в) ни один из узлов не принадлежит множеству своих потомков (нет петель.
Узел, ие имеющий потомков, называется *листом* или *концевой точкой.* Узлы, имеющие потомков, называются *внутренними* узлами дерева.)
На рис. 6 изображена часть дерева, соответствующая последовательной процедуре решения при постановке медицинского диагноза.

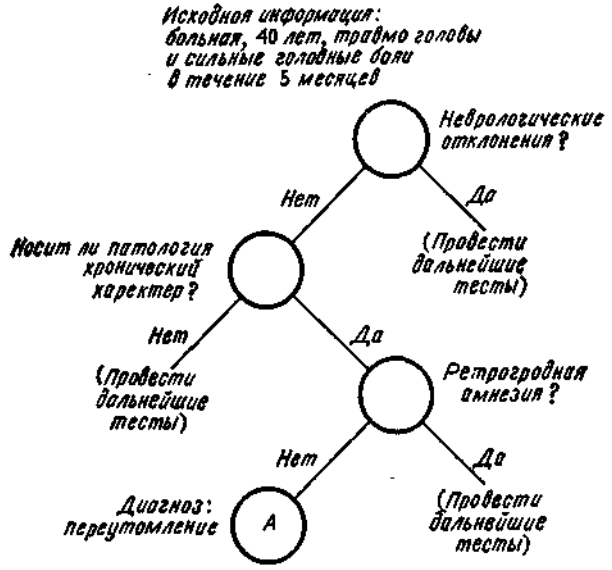


Рис.6. Пример последовательной процедуры решения при постановке медицинского диагноза. Кружки указывают путь классификации в рассматриваемом случае.

Первый тест относится к самому верхнему узлу дерева; в зависимости от результата теста следующий тест выбирается или из правого или из левого узла, расположенного ниже узла, соответствующего только что выполненному тесту. Название класса связывается с концом каждой ветви, например узел *A* на рис.6.

Если задано одно и то же множество тестов, то для выполнения последовательных процедур решения, вообще говоря, потребуется меньше тестов, чем для эквивалентной параллельной процедуры, а, значит, будет израсходовано меньше вычислительных ресурсов. С другой стороны, если есть возможность выполнить параллельную процедуру, то последовательная процедура может оказаться значительно более долгой. Очевидно, что если мы вынуждены осуществлять параллельную процедуру на последовательной машине (что обычно и бывает при использовании цифровой ЭВМ), то для выполнения соответствующей последовательной процедуры потребуется в самом худшем случае столько же времени, сколько для параллельной.
Существенный недостаток последовательной процедуры решения состоит в том, что она подвержена ошибкам в случае ненадежности отдельных тестов, как в смысле ненадежности устройства, их выполняющего, так и в том смысле, что каждая компонента x_i вектора описания определяется вероятностно при задании объекта. (Отметим, что в этих случаях применяется один и тот же формализм.) Если произошла ошибка измерения, то последовательная процедура может выбрать в дереве неверный путь, причем при этом нет возможности поправить дело. В параллельной же процедуре ошибки измерения не столь опасны, поскольку рассматриваемая классификация будет зависеть от всех имеющихся результатов испытаний.

1.2.4. Варианты описаний объектов

Третьим параметром, определяющим возможные задачи распознавания образов, являются способы описаний самих объектов. В большинстве задач классификации **объект можно считать *набором результатов измерений.*** Какова природа этих измерений и что значит их природа для процедур классификации и распознавания? Эти вопросы совсем не тривиальны. Например, в большинстве изучаемых случаев (особенно в статистике) считается, что измерения определяют *евклидово пространство описаний,* и каждый объект представляется точкой в этом пространстве. Это позволяет комбинировать измерения, чтобы определить для каждого класса местоположение „типичной" точки в пространстве описаний. Такая **процедура хорошо работает для задач измерения и группирования физических объектов.** Например, кажется разумным выработать понятие „здоровой полноты", основанное на росте и весе, и использовать его для разделения студентов на потенциальных атлетов и прочих. Разумность такого классифицирующего понятия, однако, целиком зависит от свойств, которые мы приписываем различным измерениям пространства

описаний. В данном случае идею „близости по мере (расстоянию) в евклидовом пространстве" можно интерпретировать как „подобие в росте и весе", что разумно. В других случаях нет разумной интерпретации расстояния. Например, так было бы в случае, если бы основными измерениями были определяющие признаки, выражаемые такими величинами, как пол, место рождения и раса. (На самом деле существует ряд шкал, промежуточных между шкалами, определяющими евклидово расстояние, и чисто номинальными шкалами. Большинство психологических шкал, таких, как интеллект или мужественность — женственность, является шкалами порядка. В этих шкалах объекты ранжируются относительно друг друга, однако разница между соседними рангами не определяется.) Иными словами, **описание объекта — это *список признаков,* а не набор измерений.** В любом случае объект можно представить в виде вектора описания; правда, имеющие смысл математические операции с такими векторами будут совершенно другими.

***Структурные описания* дают другой способ описания объектов.** Грубо говоря, **структурное описание выделяет взаимоотношения между компонентами объекта, а не характеристики объекта, получаемые в серии измерений. Наиболее ясный пример структурных описаний — лингвистические объекты.** При обсуждении формальных языков показано, что предложение можно эффективно описать правилами непосредственно составляющих, которые выражают отношения между элементами предложения. Без сомнения, ненужной была бы мера, задаваемая на предложениях, если бы она определялась числом появлений в них различных элементов. Какая польза была бы от утверждения, что английское предложение содержит три существительных и два глагола? Было обнаружено, что те же рассуждения можно применить и к различным областям, весьма далеким от традиционной лингвистики. Одно из наиболее интересных приложений структурного описания — использование его в анализе изображений . Например, мы не видим иного, кроме структурного, способа описания определяющих характеристик множества симметричных двумерных рисунков.

Процесс описания важен и сам по себе. Должна быть „максимально примитивная" сенсорная система для преобразования во внутренний код физических воздействий, поступающих на классифицирующее устройство. Общепринято называть это процессом *преобразования.* **Животным в качестве преобразователей служат глаза, уши и другие органы чувств** а вычислительные машины используют, например, устройства, реагирующие на изменения электрического напряжения. Назовем внутренний код, получаемый при

преобразовании, кодом *уровня* 0. Любые два физически различных объекта, производящие одинаковое кодирование уровня 0, для устройства распознавания образов будут неразличимы. При вычислениях различия уровня 0 выглядят тривиальными. Преобразование уровня 0 дает цепочку двоичных цифр, в которой каждая цифра представляет наиболее примитивное свойство вычислительной системы. С этими цепочками непосредственно работать неудобно, поэтому их обычно перекодируют в блоки, возможно, облегчающие процесс распознавания образов. Будем называть это кодированием *уровня* 1. Оно включает в себя разбиение входной последовательности на подблоки. Это вызывает дальнейшую потерю информации, поскольку не все двоичные конфигурации можно выразить с помощью буквенно-цифрового кода. Подобным же образом, только в значительно большем объеме, происходит потеря информации в биологических системах. Например, Хант и Макоус оценили уменьшение количества информации при чтении про себя с 10^9 бит в секунду на входе до 35 бит в секунду на выходе.

Результат кодирования уровня 1 состоит в получении данных, удобных для работы алгоритма распознавания образов. В биологических системах это достигается в основном путем отбрасывания ненужных данных, но почти столь же эффективной может оказаться простая перекодировка данных. Это проиллюстрировано на рис. 7, на котором „правильное" перекодирование превращает очень сложную задачу распознавания образов в очень легкую.

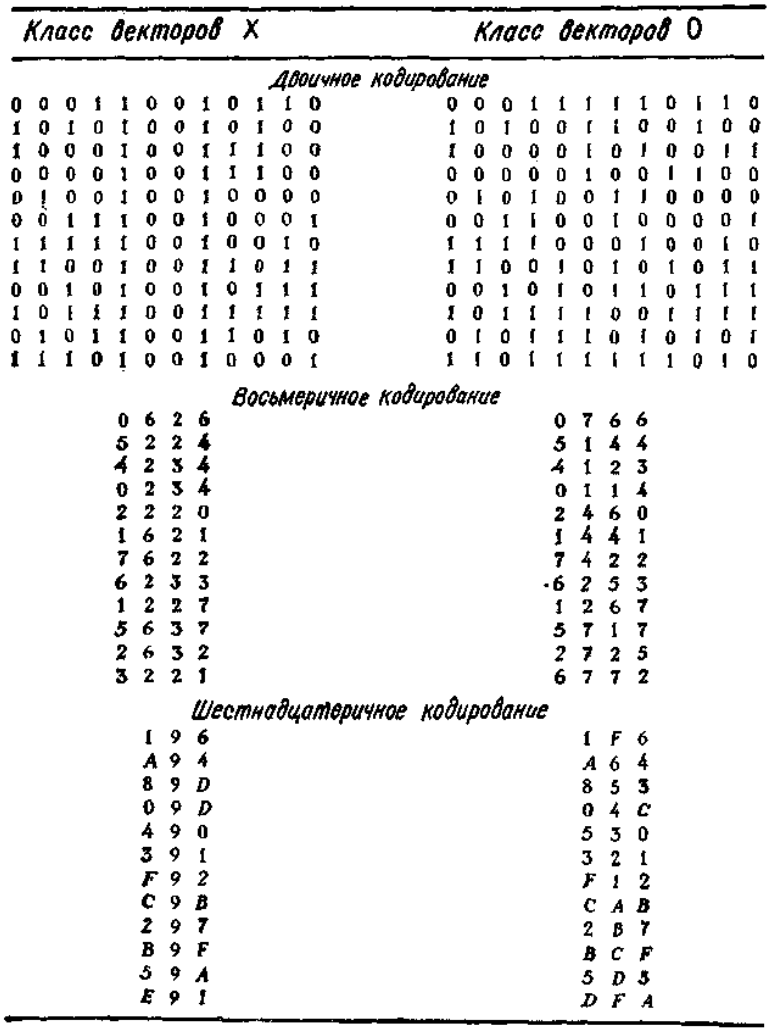


Рис.7. Результат кодирования при распознавании образов.

В верхней таблице рисунка приведены два класса двоичных векторов, обозначенные X и 0. Правило для классификации векторов найти в принципе можно, но для большинства людей оно не очевидно. В средней таблице представлены восьмеричные эквиваленты двоичных векторов. Правило классификации найти все еще тяжело. В нижней

таблице двоичные векторы опять преобразованы, на этот раз в шестнадцатеричные векторы, и ответ очевиден.
Нет ничего специального в переходе от уровня 0 к уровню 1; можно было бы легко перейти от уровня 1 к уровню 2, от уровня 2 к уровню 3 и т. д. Число переходов для устройства распознавания образов, которое ими не управляет, не имеет значения. Таким образом, если программа для ЭВМ предназначена для классификации слов и перед вводом данных в программу всегда производится преобразование разряды — символы, то не важно, будет программа оперировать с разрядами или с символами. Намного существеннее различие между программами выделения признаков, которые могут изменяться самой программой распознавания образов во время поиска решения, и теми, которые остаются неизменными. Например, рассмотренную задачу (рис.7) можно было бы решить с помощью сложного алгоритма, работающего с двоичными или восьмеричными образами, или с помощью простого алгоритма, который испытывал бы различные коды до тех пор, пока не нашел шестнадцатеричный, легко решающий задачу. Вообще мы должны различать программы распознавания образов с фиксированным кодированием данных и программы, в которых есть возможность менять вид кодирования.

1.2.5. Классификация датчиков СИИ

Главная цель информационного очувствления СИИ — сделать их способными работать в неупорядоченной и случайной обстановке. Если СИИ не снабжены информационными датчиками, то они будут не способны воспринимать какие-либо изменения в своей рабочей обстановке и реагировать на них. Чтобы СИИ могли работать эффективно, они должны снабжаться группой датчиков, и кроме того необходимо соответствующее программное обеспечение, с тем чтобы СИИ могла получать данные от датчиков и вырабатывать в реальном времени необходимые команды и информацию для принятия решения. Общая блок-схема системы искусственного интеллекта показана на рис. 8.

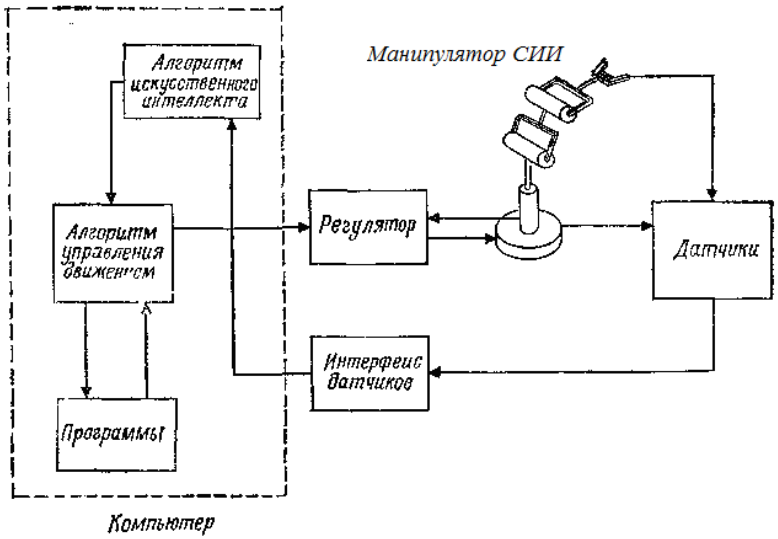


Рис. 8. Общая блок-схема системы искусственного интеллекта.

Система ИИ представляет собой многоуровневую иерархическую замкнутую систему управления. На нижних уровнях положение, скорость и, возможно, ускорение включаются в обратную связь с помощью оптических датчиков, тахометров и акселерометров соответственно. На верхних уровнях иерархического управления система обладает рядом более сложных датчиков внешней информации, которая используется в цепи обратной связи.
Вообще датчики СИИ предназначены для регулярного выполнения следующих функций.
1. Получение априорной информации о положении и (или) ориентации объектов и о рабочих деталях, а также о самой СИИ.
2. Коррекция ошибок положения и (или) ориентации объектов.
3. Получение апостериорной информации для обнаружения дефектов объектов с помощью визуальной проверки, т. е. с помощью системы технического зрения либо с помощью ультразвуковых или звуковых средств.
4. Непрерывное получение информации определяющей профиль поверхностей объектов с помощью средств тактильного очувствления.

5. Получение непрерывной информации о других аспектах окружения СИИ, таких, как наличие температурных градиентов, химических веществ и волн.
СИИ должны быть снабжены определенными датчиками ощущений, чтобы они могли успешно выполнять задания, обходясь без вмешательства человека. Некоторые из них могут быть аналогами органов чувств человека, таких, как слух, зрение, тактильная чувствительность. Однако при этом не обязательно ограничиваться человеческими органами чувств. СИИ можно обучить воспринимать радиоволны, ультразвуковые колебания, ультрафиолетовое излучение или электрические сигналы путем простого подсоединения к ее «центральной нервной системе» нужных датчиков, дающих на выходе требуемый электрический сигнал. В больнице СИИ-анестезиолог может осуществлять непосредственный контроль за дозировкой анестезирующего средства в соответствии с уровнем электрических ритмов человеческого мозга, определяемым электродами электроэнцефалографа, установленными на голове пациента.
Разработки сенсорных устройств СИИ в немалой степени стимулируются потребностью во внешних устройствах для цифровых вычислительных машин. Во всем мире разрабатываются различные виды этих устройств. Исследуются различные устройства для распознавания образов, искусственные глаза для чтения букв и цифр, искусственные уши для восприятия речи, различные виды тактильных датчиков и другие аналоги органов чувств.
Существуют органы чувств животных, которые мы до сих пор еще не научились воспроизводить. Наиболее сложными из них являются вкусовые и обонятельные.
Было бы удобно, если бы органы чувств СИИ обладали всеми свойствами органов чувств человека и животных. Весьма полезным является, например, свойство аккомодации. Как только происходит внезапное изменение в стимуляции нервной клетки жниотного, резко возрастает выход нервных импульсов в нервную систему. Если, однако, дальнейшая стимуляция сохраняет неизменную величину, нервная активность падает. По существу эта активность представляет собой форму квазидифференциации реакции при изменении воздействия. Такой вид активности, по-видимому, основной в нервной системе животных, причем он не сводится только лишь к временной области. Например, в сетчатке глаза имеется некая форма пространственной квазидифференциации, которая обеспечивает эффект выделения контуров изображения, проецируемого на сетчатку. Желательно, чтобы такая способность к временной и

пространственной дифференциации была введена в число свойств датчиков нервной системы ИИ.
Еще одно желательное для СИИ свойство сенсоров нервной системы животных состоит в том, что сигнал на их выходе не прямо пропорционален величине стимуляции, а пропорционален ее логарифму. Этот закон (известный как закон Вебера—Фехнера) позволяет сенсорам животных работать в очень широком диапазоне интенсивностей стимуляции, и поэтому подчинение ему желательно для нервной системы ИИ. Следует также наделять СИИ внутренним чувством времени. Эксперименты по отключению сенсоров человека позволяют утверждать, что человек, по существу, всецело судит о времени по внешним событиям и что при отсутствии на входе информации от внешних источников утрачивается всякое «чувство времени». Если это так, то нетрудно оснастить СИИ времяхранящим устройством, что будет весьма полезно, поскольку тогда СИИ можно будет отдавать распоряжение о выполнении определенного действия в определенное время.
Ниже мы представим структурную классификацию датчиков СИИ. Принято классифицировать датчики СИИ на две различные, но широкие группы, а именно внутренние и внешние, как показано на рис. 9.

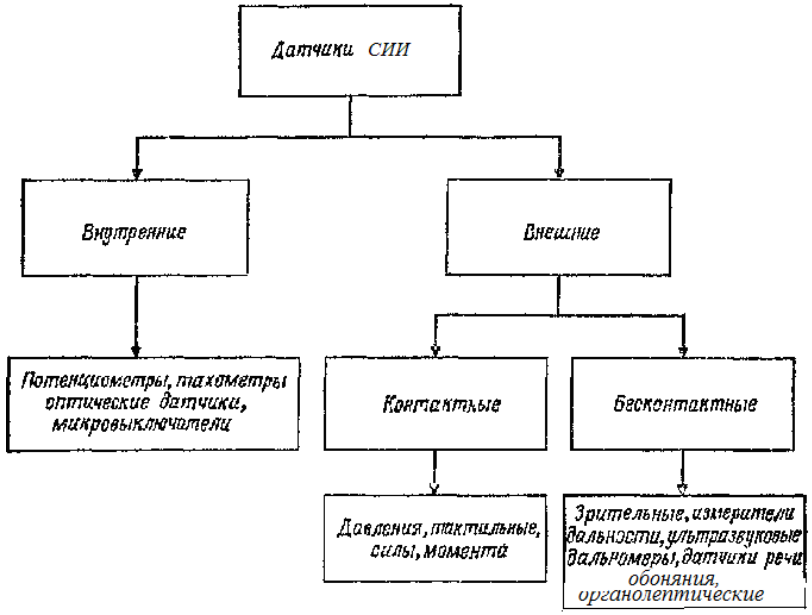


Рис.9. Классификация датчиков СИИ

Внешние датчики далее подразделяются на контактные и бесконтактные. Простым тактильным датчиком является микровыключатель, который улавливает наличие барьера, препятствий либо поверхностей. Таким образом, им можно пользоваться для того, же измерять размеры предметов. Датчики сил и моментов измеряют силы и моменты. Они могут состоять из простых измерителей деформации, либо это могут быть пьезоэлектрические преобразователи.
Другой важной группой датчиков СИИ являются бесконтактные датчики; к ним относятся визуальные, локационные и акустические датчики, измерители дальности, датчики температуры и химические датчики.
Дадим краткое описание некоторых бесконтактных датчиков, указав их преимущества и недостатки.
Начнем с визуальных датчиков, заметим, что они в основном базируются на цифровых ТВ-камерах или сканирующих устройствах с лазерным лучом. Оцифрованные сигналы из камер с частотой примерно 50 кадров в секунду поступают в центральную ЭВМ, которая в свою очередь анализирует данные и извлекает необходимую и полезную информацию. Этот процесс более подробно рассматривается ниже в этой главе. Здесь же мы остановимся на другом классе бесконтактных датчиков, а именно на классе локационных датчиков. Датчики этого типа обнаруживают наличие предмета в пределах ограниченного рабочего пространства около самих датчиков. Хорошим примером локационных датчиков является индуктивный датчик, с помощью которого можно обеспечивать постоянное расстояние между объектами. Другим примером простого локационного датчика является светодиод и соответствующий светоулавливающий диод (фотодиод), который может принимать отраженный от поверхности световой луч и таким образом сигнализировать о ее близости. Интенсивность принимаемого сигнала в большой степени зависит от отражающей способности поверхностей, расстояние до которых нужно измерить, что является недостатком таких датчиков. Акустические датчики включают такие устройства, как специальные микрофоны, способные воспринимать отдельные слова либо определять наличие изъянов и трещин в материалах.
Другим способом бесконтактного измерения является определение расстояния до объекта, на основании которого можно получить информацию о форме предметов или распределении объектов в пространстве такими методами, как триангуляция. Более подробно это рассматривается ниже в этой главе.

Существует два основных способа использования датчиков СИИ: включение датчика в цепь обратной связи при подвижно СИИ и использование датчиков при неподвижной СИИ. При использовании первого способа СИИ управляется датчиком во время самого движения. Например, большинство зрительных систем работает в режиме обратной связи, так что камера следит за ошибкой между реальным и желаемым положениями объектов. Компьютер пользуется этой информацией для того, чтобы эффективно перемещать манипулятор с к ТВ-камерой для достижения требуемого положения ТВ-камеры. Главной трудностью этого метода является большое время вычислений для анализа полученных изображений и соответственно управления СИИ для коррекции его кинематического положения. При втором способе использования датчика СИИ остается неподвижной, тогда как датчики воспринимают окружающую обстановку и осуществляют обратную связь (рис.10).

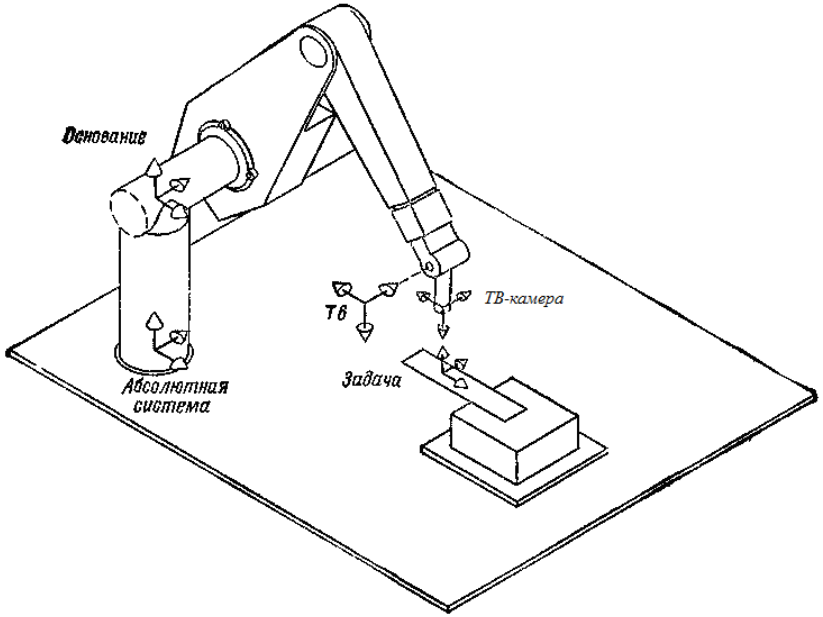


Рис.10. Статическое очувствление движения СИИ

Рассмотрим пример.
Предлагается рассмотреть систему технического зрения (СТЗ), включенную в цепь обратной связи (рис.11).

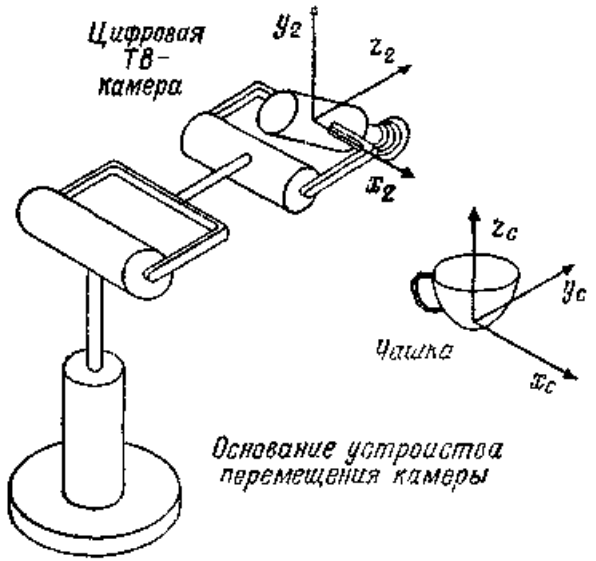


Рис.11. Зрительная система СИИ.

Камера наблюдает за движением предмета, будучи прикрепленной к звену 5 манипулятора. Матрица преобразований ${}^5\mathbf{C}$, описывающая положение системы координат камеры относительно системы $\mathbf{T}_5$, и матрица преобразований, описывающая положение системы координат схвата относительно системы $\mathbf{T}_5$, т. е. ${}^6\mathbf{A}_5$, имеют вид

$$ {}^5\mathbf{C} = \begin{bmatrix} 0 & 0 & -1 & 8 \\ 0 & -1 & 0 & 0 \\ -1 & 0 & 0 & 5 \\ 0 & 0 & 0 & 1 \end{bmatrix}, \tag{1} $$

$$ \mathbf{A}_5^6 = \begin{bmatrix} 0 & -1 & 0 & 0 \\ 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 10 \\ 0 & 0 & 0 & 1 \end{bmatrix}. \tag{2} $$

Система координат предмета связана с системой координат камеры преобразованием ${}^C\mathbf{O}$. Система координат предмета связана с системой координат схвата преобразованием ${}^6\mathbf{O}$. Нужно определить обобщенный дифференциальный вектор, связывающий системы координат схвага и

предмета, если обобщенный дифференциальный вектор, связывающий системы координат предмета и камеры, имеет вид

$$^{C}\mathbf{D}=[-0.2 \quad 0.2 \quad 0 \quad 0 \quad 0 \quad 0.2]^{T}. \tag{3}$$

Рассмотрим рис. 11. Отметим, что система координат схвата связана с системой координат камеры $^{C}\mathbf{G}$ следующим образом:

$$^{c}\mathbf{G} = {}^{5}\mathbf{C}^{-1}\mathbf{A}_{5}^{6}. \tag{4}$$

Заметим, что

$$^{5}\mathbf{C}^{-1} = \begin{bmatrix} 0 & 0 & -1 & 5 \\ 0 & -1 & 0 & 0 \\ -1 & 0 & 0 & 8 \\ 0 & 0 & 0 & 1 \end{bmatrix}. \tag{5}$$

Следорательно, $^{C}\mathbf{G}$ имеет вид

$$^{c}\mathbf{G} = \mathbf{CAM}^{-1}\mathbf{A}_{6} = \begin{bmatrix} 0 & 0 & -1 & 5 \\ 0 & -1 & 0 & 0 \\ -1 & 0 & 0 & 8 \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} 0 & -1 & 0 & 0 \\ 1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 10 \\ 0 & 0 & 0 & 1 \end{bmatrix}, \tag{6}$$

или

$$^{c}\mathbf{G} = \begin{bmatrix} n_x & o_x & a_x & p_x \\ n_y & o_y & a_y & p_y \\ n_z & o_z & a_z & p_z \\ 0 & 0 & 0 & 1 \end{bmatrix} = \begin{bmatrix} 0 & 0 & -1 & -5 \\ -1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 8 \\ 0 & 0 & 0 & 1 \end{bmatrix}. \tag{7}$$

Таким образом, $\mathbf{n} = 0\mathbf{i} - 0\mathbf{j} + 0\mathbf{k}$, $\mathbf{o} = 0\mathbf{i} + 0\mathbf{j} + \mathbf{k}$, $\mathbf{a} = \mathbf{i} + 0\mathbf{j} + 0\mathbf{k}$ и $\mathbf{p} = 5\mathbf{i} + 0\mathbf{j} + 8\mathbf{k}$. Заметим таже, что

$$^{c}\mathbf{d} = -0.2\mathbf{i} + 0.2\mathbf{j} + 0\mathbf{k}, \tag{8}$$

$$^{c}\boldsymbol{\delta} = 0\mathbf{i} + 0\mathbf{j} + 0.2\mathbf{k}. \tag{9}$$

Следовательно,

$$^{T}d_x = \boldsymbol{\delta} \cdot (\mathbf{p} \times \mathbf{n}) + \mathbf{d} \cdot \mathbf{n} = \mathbf{n} \cdot (\boldsymbol{\delta} \times \mathbf{p} + \mathbf{d}), \tag{10}$$

$$^{T}d_y = \boldsymbol{\delta} \cdot (\mathbf{p} \times \mathbf{o}) + \mathbf{d} \cdot \mathbf{o} = \mathbf{o} \cdot (\boldsymbol{\delta} \times \mathbf{p} + \mathbf{d}), \tag{11}$$

$$^{T}d_z = \boldsymbol{\delta} \cdot (\mathbf{p} \times \mathbf{a}) + \mathbf{d} \cdot \mathbf{a} = \mathbf{a} \cdot (\boldsymbol{\delta} \times \mathbf{p} + \mathbf{d}), \tag{12}$$

$$^{T}\delta_x = \mathbf{n} \cdot \boldsymbol{\delta}, \tag{13}$$

$$^{T}\delta_y = \mathbf{o} \cdot \boldsymbol{\delta} \tag{14}$$

$$^{T}\delta_z = \mathbf{a} \cdot \boldsymbol{\delta}, \tag{15}$$

так что $\boldsymbol{\delta} \times \mathbf{p} = 0\mathbf{i} + 1\mathbf{j} + 0\mathbf{k}$.

Следовательно,

$$^{6}\mathbf{d} = -1.2\mathbf{i} + 0\mathbf{j} + 0\mathbf{k} \tag{16}$$

$$\boldsymbol{\delta} = 0\mathbf{i} + 0.2\mathbf{j} + 0\mathbf{k}. \tag{17}$$

Наконец,

$$^{6}\mathbf{D} = [-1.2 \quad 0 \quad 0 \quad 0 \quad 0.2 \quad 0]^{T}. \tag{18}$$

1.2.6. Сенсорные системы СИИ

1.2.6.1. Сенсорные системы человека

Анализ сенсорных систем человека представляет интерес для создателя СИИ, так как он может встретиться с необходимостью как можно более полно воспроизвести их функции в СИИ. Это бывает необходимо, например, в тех случаях, когда непосредственно на СИИ возлагается решение задач, которые обычно решает человек. Осязанию человека соответствуют четыре отчетливых ощущения, и кожа человека содержит, по-видимому, отдельные рецепторы для каждого из них. Ими являются ощущения холода, тепла, боли и давления. Чувствительность различных участков кожи к этим ощущениям различна, и поверхностная плотность распределения различных типов рецепторов з различных частях тела не одинакова. Осязание, по-видимому, должно быть важнейшей функцией для СИИ. Ощущение холода и тепла не имеет такого значения для сохранения жизни СИИ, как для животного, хотя оно и может играть важнейшую роль, если на СИИ возложить, например, задачу поддержания в допустимых пределах жизненно важных для человека параметров. Восприятие давления является функцией, без которой СИИ не может обойтись. Эта функция может вводиться в СИИ различными методами.

1.2.6. 2. Реакция рецепторных нервных клеток на воздействия

Переходная реакция рецепторных нервных клеток животных хорошо изучена; совершенно очевидно, что она представляет собой реакцию с опережением по фазе. При постоянной величине воздействия частота импульсов нервной клетки сначала резко возрастает до высокого уровня, а затем экспоненциально спадает до низкого уровня.

Исходное значение частотно-временной зависимости, как и ее установившийся уровень, зависит от величины раздражения. В таком виде переходная реакция показана па рис. 12.

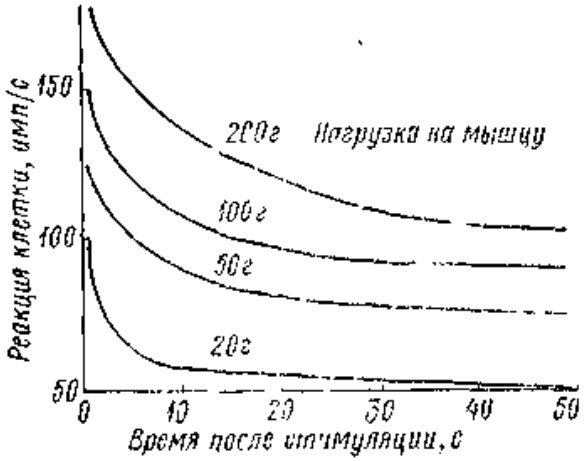


Рис. 12. Переходная характеристика рецепторных нервных клеток

Тем не менее возможность получения частотной характеристики рецепторной нервной клетки вызывает сомнения, связанные с нелинейностью и трудностью определения выхода, выраженного в виде числа импульсов в секунду, поскольку выходной сигнал нерегулярен по частоте. Однако типичная реакция определена и имеет следующий вид:

$$\frac{27\,(1+30p)}{(1+5p)\,(1+0{,}35p)}.$$

Эта реакция с опережением по фазе в 20° обеспечивается при отношении граничных частот диапазона 10:1 и при средней частоте диапазона 0,1 Гц. В рецепторах СИИ имеет смысл предусмотреть некоторое опережение по фазе, хотя величина его будет зависеть от системы управления в целом, рассматриваемой как замкнутый контур с внешней средой в цепи обратной связи.

1.2.6.3. Количество нейронов у человека

Поскольку СИИ предназначена для моделирования функционирования, по крайней мере, некоторых сторон нервной

деятельности человека, полезно привести число нейронов у человека (табл. 1).

Таблица 1

Вид восприятия	Сенсорные входы: Рецепторы	Сенсорные входы: Нервные волокна	Пропускная способность канала
Зрение	$2\cdot10^8$	$2\cdot10^6$	$5\cdot10^7$
Слух	$2\cdot10^4$	$2\cdot10^4$	$4\cdot10^4$
Осязание	$5\cdot10^5$	$1\cdot10^4$	—
Боль	$3\cdot10^6$	—	—
Тепло	$1\cdot10^4$	$1\cdot10^6$	—
Холод	$1\cdot10^5$	—	—
Запах	$1\cdot10^7$	$2\cdot10^3$	—
Вкус	$1\cdot10^7$	$2\cdot10^3$	—
Всего	$3\cdot10^8$	$3\cdot10^6$	—

Необходимо отметить, что эти величины не могут быть определены с большой степенью точности, поэтому приводятся здесь скорее для сравнения, чем для непосредственного научного использования. Мы не ставим себе целью дать точные обобщенные данные; заметим только, что произведение максимальной пропускной способности мозга в 50 бит/с на среднюю продолжительность жизни, скажем 70 лет, дает величину 10^{10}, т. е. того же порядка, что и количество клеток в мозге. Отметим, что в среднем на каждое нервное волокно приходится около 100 рецепторов и 3000 нейронов центральной нервной системы. Кайдель считает, что пропускная способность организма человека составляет 10^7 бит/с.

Требуя согласованной работы нервных волокон, мы значительно увеличиваем вероятность того, что выходной сигнал действительно соответствует некоторому входному сигналу, а не просто шуму или помехе. За определенность организм расплачивается информацией. Глаз передает мозгу лишь сотую долю информации, которую получает. Однако вероятность того, что передаваемая в мозг информация носит

случайный характер, ничтожно мала и равна 2^{-100}. В этом и состоит преимущество отсеивания информации в отношении 100 : 1 при передаче ее от глаза в мозг.
Приведем следующее сравнение между зрительной н слуховой системами человека:

	Зрительная система	Слуховая система
Количество рецепторов на1 волокно . .	130	1
» нейронов в первичной коре	$538 \cdot 10^6$	$100 \cdot 10^e$
» волокон на 1 клетку коры	1 : 538	1 : 3000

Интерес представляет также количество различных уровней сенсорных сигналов, которые могут поступать в центральную нервную систему от тех или иных периферических рецепторов и различаться человеком. Это количество, как правило, изменяется от 3 до 9, хотя возможны и большие вличиыы.
Используется несколько видов протезных сенсорных устройств. В качестве примера можно привести алфавит Брейля для общения со слепыми и вибротактильные устройства для связи через рецепторы кожи.

1.2.6.4. Рефлекторное действие СИИ

Основной задачей рефлекторной деятельности человека и животного является самосохранение. Желательно наделить СИИ подобной рефлекторной способностью, обеспечивающей ее жизнеспособность. Например, тепловые датчики , быстро срабатывая могут воздействовать непосредственно на конечностно-двигательную систему ИИ. Это позволит при необходимости включить рефлекс отвода конечности и обеспечить ее защиту от повреждения. Для такого рефлекторного действия нет необходимости использовать путь, проходящий через всю нервную систему, ибо для защиты необходима быстрая реакция. Показано, что рефлекторную деятельность такого рода можно наделить способностью к ассоциированию в центральной нервной системе СИИ с другими, нерефлекторными входами центральной нервной системы.
Тем не менее на рефлекторную деятельность СИИ накладывается важное ограничение, которого нет в организме животного. Необходимо предусмотреть, чтобы рефлекторные действия СИИ не причиняли вреда человеку. Пусть уж лучше рефлекторное действие нанесет повреждение СИИ, чем человеку.

1.2.6.5. Усиление контраста

Исследования показывают, что нервная система человека и животного способна различными путями усиливать эффект изменения раздражения относительно некоторого постоянного раздражителя. Например, внезапное изменение времени раздражения рецепторной клетки сопровождается усилением нервной активности, в частности резким возрастанием частоты нервных импульсов. Однако, если в дальнейшем раздражение остается неизменным, нервная активность понижается, как это показано на рис. 12.
Одним из видов подобного усиления является усиление контраста во времени. Аналогичным образом происходит, по-видимому, пространственное усиление контраста в тех частях нервной системы, которые воспринимают пространственную активность. Так, сетчатка глаза, вероятно, более чувствительна к краям объекта, чем к его поверхностям, и к внезапным переменам в освещении, чем к ровному комнатному освещению.
Нервная система вообще реагирует только на изменения в раз-дражении. При изменениях раздражения во времени наблюдается эффект, очень напоминающий то, что происходит в схемах опережения по фазе, применяемых в технических системах управления.

1.2.6.6. Обнаружение газа и влаги

Довольно сложно воспроизвести человеческое обоняние техническими средствами. Однако некоторые газы можно обнаруживать. Так, при соприкосновении сгораемой смеси газа и воздуха с некоторыми катализаторами, например платиной или палладием, выделяется тепло и изменяется электрическое сопротивление катализатора, что можно обнаруживать прямым измерением. Подобные детекторы, безусловно, очень важны для предотвращения пожара и взрыва. Кислород можно обнаруживать, используя его парамагнитные свойства, а некоторые газы — используя их теплопроводность, в устройстве, называемом термокондуктометрическим детектором. Для обнаружения различных газов можно использовать также инфракрасное поглощение. Водяные пары обнаруживаются гигрометрами различных типов. Тем не менее ни один из этих методов но является идеальным для применения в СИИ. Следует ожидать, что исследования в этой области приведут к появлению более совершенных детекторов запаха.

Человеческий вкус также является ощущением, которое мы еще не можем достаточно достоверно воспроизвести. Лучше всего использовать для этих целей измерители pH.
Человек способен ощущать и воспринимать четыре основных вкусовых качества (сладкое, горькое, кислое и соленое), которыми в разной степени наделены различные вещества.
Наибольший объем вкусовой и обонятельной информации поступает при небольшой вероятности конкретного вкусового качества или запаха. Хорошо замечается то, что незнакомо.
Райт считает, что нервная система животного обнаруживает запахи, улавливая колебания молекул в дальней инфракрасной области.
Газы обладают способностью изменять цвет различных химических веществ, что часто используется в газовом анализе. Можно взять различные реактивы, и припускать через них газ. Следует отметить, что у животных нервные окончания, предназначенные для обнаружения запаха, по-видимому, быстро погибают и очень часто обновляются. В Японии разработано газоулавливающее устройство, изготовленное из оксидированных металлов, таких как окись олова, окись цинка и полуторная окись железа. В этом устройстве производится очень значительное, хотя и обратимое, уменьшение электрического сопротивления при соприкосновении с газами-восстановителями: водородом, окисью углерода, метаном, пропаном, спиртом, эфирным маслом и ацетиленом.
Твердотельная технология позволила изготовить кислородный анализатор, который может применяться, например, для определения концентрации кислорода в топочных газах и, следовательно, для контроля интенсивности горения. Прибор содержит стабилизированный циркониевый элемент, работающий при температуре 850° С и генерирующий напряжение, которое изменяется по логарифмическому закону в зависимости от разности между парциальными давлениями кислорода и контрольного источника. Как сообщалось, точность этого прибора ±0,1%, время срабатывания 0,2 с в диапазоне температур 10— 760° С, выходной сигнал 4—20 мА (или 1—5 В).
Для восприятия влажности применяются различные элементы, в том числе хлористо-литиевые (датчик «Данмор»), углеродные, элементы на базе полиэлектролитного сопротивления, керамические элементы, емкостные устройства и элементы на базе окиси алюминия. Все они в той или иной степени нестабильны благодаря ионному загрязнению, растворимости в воде, поляризации, химическому и механическому разрушению. Тома использовал гибкую ленту, выполненную из пятислойной пленки бутирата ацетилцеллюлозы, который как стало

известно, дает высокую чувствительность при химической и механической стабильности. Это химическое вещество используется в качестве элемента в среде тщательно очищенной двуокиси углерода, а для повышения чувствительности элемент подвергается воздействию водного раствора едкого натра. В результате достигаются сопротивление около 2500 Ом и работоспособность при относительной влажности 10—90% с постоянной времени свыше 100 с. При этом сопротивление почти не зависит от напряжения и температуры. Весьма вероятно, что подобные разработки можно будет использовать в СИИ.

1.2.6.7. Регулирование температуры

Температура оказывает воздействие на механизм управления всеми системами животного. Внутренняя температура хладнокровного (пойкилотермного) существа изменяется вместе с изменением окружающей температуры, и поэтому уровень активности его организма до некоторой степени ограничен окружающими условиями. Теплокровные животные, напротив, наделены автоматической системой терморегулярии, которая поддерживает внутреннюю температуру животного на более или менее постоянном уровне. Минимально допустимая температура тела равна приблизительно 37° С; ниже этой границы ферментная активность организма значительно падает. С другой стороны, температура, превышающая приблизительно 41° С, влечет необратимые изменения в клетках центральной нервной системы. Из этого следует, что температура тела нуждается в довольно жестком регулировании.
К счастью, СИИ совсем не нужны те многочисленные средства, которые живые организмы приобрели в процессе эволюции для терморегуляции. Правда, некоторое регулирование температуры необходимо, но требования при этом не очень жестки. Так, есть полупроводники, которые работают в диапазоне температур окружающей среды от —50 до +150° С. Нередко возникает потребность в системе охлаждения, поскольку все электрические и электронные схемы управления выделяют тепловую энергию. Подобные случаи часто встречаются при использовании компактного оборудования, например интегральных электронных схем. Однако из-за отсутствия жестких температурных ограничений потребность в сложных системах терморегулирования возникает редко; обычно вполне достаточно иметь системы охлаждения, использующие конвекционные потоки в окружающем воздухе.
Следует отметить, что терморегуляция в организме животного определяется тем, что он на 70% состоит из воды. Благодаря этому

изменение температуры внешней среды значительно меньше влияет на температуру тела, чем было бы при любом ином составе. Скрытая теплота замерзания воды имеет одно из самых высоких значений, за счет чего она регулирует температуру Земли, поскольку нагревание уменьшает, а охлаждение увеличивает ее ледяной покров. Подобным образом поддерживается постоянство температуры электронных компонентов, например кварцевых кристаллов. Скрытая теплота парообразования воды также относительно велика, и это обстоятельство используется в системах охлаждения за счет испарения воды. Точка замерзания воды приблизительно на 100° С выше критической температуры многих часто встречающихся газов. Из сказанного ясно, что наличие воды в организме животного является очень полезным фактором.

В ряде случаев СИИ необходимо наделить способностью к измерению уровня температуры либо внутри себя самого, либо к окружающей среде. Для этого можно использовать любой из хорошо известных методов электрического определения температуры. Так, при необходимости точных измерений можно использовать термосопротивления, хотя они дают недостаточно большой выходной сигнал.

Изготовляют быстрореагирующие приборы на основе кремния. Следует упомянуть об одном из способов изготовления термостолбика, состоящего из множества последовательных термопарных соединений. Для этого используется константановая проволока, на которую наносятся полоски меди. Поскольку медь обладает лучшей проводимостью, в областях с медным покрытием большая часть тока проходит через медь, хотя в других местах весь ток проходит через константан. Таким образом достигается эффект большого количества последовательных соединений в устройстве, которое оказалось очень удобным для определения теплового потока.

1.2.6.8. Регулирование температуры с использованием термисторов

Если СИИ должна осуществлять точное регулирование температуры, можно применять такие устройства, как термометры сопротивления, подключенные к усилителям. Однако во многих случаях не требуется исключительно высокая точность регулирования температуры, а желательно максимальное упрощение необходимого оборудования. При этом весьма целесообразно использование термистора, который представляет собой полупроводниковый резистор, обладающий очень

высокой величиной температурного коэффициента сопротивления, равной, как правило, —4% на 1° С при 20° С.

Сопротивление термистора R экспоненциально изменяется с изменением температуры и может быть выражено формулой

$$R = A \exp(B/T),$$

где A — константа; T — температура термистора, К; B — константа, равная обычно 2500—3000 К, Дифференцированием находим

температурный коэффициент сопротивления = $-AB/T^2$.

Таким образом, при повышении температуры температурный коэффициент сопротивления уменьшается.

Часто бывает желательно, чтобы изменение сопротивления в зависимости от температуры, по крайней мере в пределах ограниченного диапазона температур, носило линейный характер. С этой целью в схему вводят параллельное сопротивление постоянной величины, а для получения ограниченной области изменения линейной проводимости схему дополняют последовательным сопротивлением. Необходимое параллельное сопротивление R_p можно найти из выражения

$$R_p = R\left(\frac{B - 2T_i}{B + 2T_i}\right),$$

где T_i—требуемое среднее значение рабочей температуры. При температуре T_i наблюдается перегиб кривой зависимости результирующего сопротивления от температуры. В этой точке скорость изменения сопротивления максимальна и равна

$$\left.\frac{dR_p}{dT}\right|_{T_i} = -R\frac{(B - 2T_i)^2}{4BT_i^2}.$$

Последовательное сопротивление для получения ограниченной области изменения проводимости находится аналогичным образом из выражения для последовательной проводимости

$$G_s = G\left(\frac{B + 2T_i}{B - 2T_i}\right)$$

и в точке перегиба

$$\left.\frac{dG_s}{dT}\right|_{T_i} = G\frac{(B + 2T_i)}{4BT_i^2},$$

где G — проводимость термистора при температуре T_i.

Таким образом, достижима линейность в диапазоне температур, превышающем ± 20° С. Аналогичный линейный диапазон изменений тока можно получить при помощи термисторных мостовых схем, для построения которых разработано множество практических методов. Вместо термисторов в качестве термочувствительного элемента можно использовать полупроводниковый диод. Еще более высокую

температурную чувствительность можно получить, используя транзистор с разомкнутой базой.
Широкий диапазон линейности напряжения (или сопротивления) термистора достигается за счет шунтирования его последовательным соединением термистора и резистора. Линейность в пределах доли градуса достигается в широком диапазоне температур от —30 до +100° С.

1.2.6.9. Датчики силовой обратной связи

Часто бывает желательно, чтобы управляющая сервосистема вырабатывала сигнал обратной связи, соответствующий механическому усилию, развиваемому на выходе.
Видимо, лучшим устройством для обеспечения сигнала обратной связи является тензодатчик, построенный на кристаллах кремния, которые создают незначительный гистерезисный эффект.
Для типичного устройства требуется сигнал порядка 2мВ/Н, который можно получить, используя мостик Уитстона, состоящий из двух тензодатчиков на кремнии *p*-типа проводимости, которые установлены на противоположных концах бруска, сделанного из мягкой стали и имеющею постоянный момент. Такое устройство работает на постоянном токе и дает линейное соотношение между создаваемой деформацией и выходным напряжением.За счет небольшого смещения устройства в исходном положении обычно создается легкий прижим порядка 2 Н. Для защиты тела тензодатчика от чрезмерного растяжения можно установить переходное устройство. Если движение в позицию захватывания происходит на большой скорости, необходимо учитывать кинетическую энергию движения и прикладывать обратный момент для сокращения времени торможения.
В качестве гидростатического манометра можно использовать диод Зенера, что дает возможность создавать малогабаритные устройства для применения в СИИ.
В одном из образцов искусственной кисти использовалось множество губчато-угольных накладок и кожа перчатки кисти также была подбита губчатой резиной, так что давление во всех точках передавалось на накладки. Накладки-датчики дают линейную реакцию в определенном диапазоне усилий, верхняя граница которого однако ограничена.
Желательно поместить датчики усилий в запястье кисти робота, с тем чтобы кисть по существу выполняла функции своеобразною весового прибора. Полученную информацию можно использовать для обеспечения движения руки, а также в аварийных ситуациях.

Целесообразно предусмотреть также дополнительные датчики, которые обычно не подвергаются нажатию, но могут нажиматься человеком в аварийных условиях для снятия давления.
В настоящее время испытывается потребность в новых типах силовых датчиков. Они должны обладать высоким уровнем выходного напряжения, быть прочными, нечувствительными к температурным изменениям и иметь небольшую массу. Возможным вариантом датчика, отвечающим некоторым из этих требований, является датчик с высокочувствительным пьезокристаллическим элементом, усилие к которому передается через жидкость, содержащуюся в той же оболочке, что и датчик.
Независимо от вида силовых датчиков обратной связи их необходимо наделить соответствующей фильтрующей способностью для устранения сигналов, зависящих от таких внешних факторов, как сильный шум и вибрация.
В некоторых устройствах важно защитить систему управления от повреждения, которое может быть причинено ей при случайном выходе из строя тензодатчика. Для этого применяются как электрические, так и механические методы защиты, а в некоторых случаях сочетание обоих методов.
Датчики осязания у человека очень чувствительны и многочисленны, что позволяет использовать их для различения формы. Существует потребность в датчиках очень малых размеров (нанодатчиков) и по возможности в интегральном исполнении, что позволит применять их в схемах, подобных тем, которые необходимы для сетчатки глаза.

1.2.6.10. Позиционная обратная связь

Применение управления с замкнутым контуром регулирования значительно снижает требования, предъявляемые к точности. В той или иной форме такая обратная связь обычна для различных систем животного.
Существует множество различных видов датчиков положения.
Для точного измерения положения общепринят цифровой вид измерений. В этих целях используется множество различных методов, реализуемых устройствами, которые можно грубо разделить на две категории.
К первой категории относятся устройства, измеряющие абсолютное положение по отношению к постоянной точке отсчета. Вторая категория включает различные виды вычислительных устройств, которые суммируют небольшие равномерные приращения для получения абсолютной величины положения. Устройства первой

категории дают большую погрешность при потере одной или более значащих цифр отсчета. Устройствам второй категории свойственна постоянная потеря информации в случае пропуска даже нескольких приращений. Эти недостатки можно устранить путем частой подстройки нуля измерительного прибора, поскольку абсолютную достоверность никогда нельзя гарантировать.

Столь точные устройства для измерения положения, как правило, громоздки и дороги и поэтому непригодны для применения в СИИ. При наличии у СИИ обратной связи за счет какого-либо «зрительного» устройства необходимость в абсолютной точности значительно уменьшается и заменяется необходимостью в точном сравнении положений.

Следует отметить, что, хотя человек и животные используют визуальную обратную связь, у них имеется в той или иной форме еще и внутренняя обратная связь, ибо в противном случае уменьшилась бы точность позиционирования при закрытых глазах или у слепых. Существует мнение, что обратная связь в нервной системе не является частью контура управления по положению, а определяет, возможно, конкретное соотношение между напряженностью и длиной мышцы.

Метод точной индикации положения, предложенный Янгом, основан на вращении стального сегмента для электромагнитного индуцирования импульсов в катушке и сравнения временных положений этих импульсов с временными положениями других, эталонных импульсов. При устранении температурного дрейфа за счет правильного монтажа этот метод может обеспечить высокую точность. Чтобы избежать необходимости использования вращающихся деталей, автор применил линейно перемещающееся поле, создаваемое многофазной обмоткой.

1.2.6.11. Генерирование магнитных импульсов

Во всех областях применения СИИ важно предусмотреть, чтобы размеры и масса всех ее механических измерительных устройств были минимальными. В методе измерения, предложенном Янгом, используется генерирование импульсов в катушке, окружающей разомкнутую магнитную цепь, при перемещении железного сердечника. Поскольку магнитный поток, пронизывающий катушку, изменяется из-за переменного сопротивления магнитной цепи, в катушке возникает э. д. с.

Мгновенное значение э. д. с. *e,* индуктируемой в катушке, состоящей из *w* витков и охватывающей магнитную цепь с изменяющимся потоком Ф, можно найти из формулы

$$e = -w\frac{d\Phi}{dt}.$$

При постоянной намагничивающей силе магнитной цепи F и изменяющемся магнитном сопротивлении R_M

$$e = -w\frac{d(F/R_M)}{dt} = \frac{wF}{R_M^2}\frac{dR_M}{dt}.$$

При наличии в магнитной цепи одной неподвижной части и одной перемещающейся

$$R_M = R_{M_1} + l/(S\mu),$$

где R_{M1} — магнитноесопротивление неподвижной части; l — длина перемещающейся части; S — площадь поперечного сечения перемещающейся части; μ — магнитная проницаемость перемещающейся части. Отсюда

$$e = \frac{wF}{[R_{M_1} + l/(S\mu)]^2}\frac{d[l/(S\mu)]}{dt},$$

так что

$$e = \frac{wF}{[R_{M_1} + l/(S\mu)]^2}\left[\frac{S\mu\,(dl/dt) - l\,(dS\mu/dt)}{S^2\mu^2}\right].$$

Если длина l мала по сравнению с $R_{M1}S\mu$, то из приведенной формулы следует, что

$$e = \frac{wF}{R_{M_1}S\mu}\left(\frac{dl}{dt} - \frac{l}{S\mu}\frac{dS\mu}{dt}\right).$$

В этой формуле член dl/dt — скорость увеличения или уменьшения длины перемещающихся частей магнитной цепи. Следовательно, для получения максимальной величины индуктированной э. д. с. e магнитная цепь должна строиться так, чтобы площадь S возрастала при уменьшении длины l и наоборот.

Это означает, чго скорость изменения магнитного сопротивления должна быть максимально высокой. Как и следовало ожидать, для получения большой величины индуктированной э. д. с. необходимы большое число витков, большая величина намагничивающей силы, высокая скорость изменения магнитного сопротивления и малое значение магнитного сопротивления неподвижной части магнитной цепи. Кроме того, величина произведения $S\mu$ должна быть небольшой, добиться из-за большой скорости изменения магнитного сопротивления.

Значение данного метода состоит в том, что его можно применять либо для измерения положения, либо для измерения скорости и что при

тщательности разработки можно добиться почти полной независимости измерений от условий окружающей среды.
Необходимая для реализации метода электронная схема может быть выполнена очень небольшой и легкой при использовании интегральных компонентов.

1.2.6.12. Резистнвные тензодатчики

При разработке СИИ важно доводить размеры и массу устройств измерения положения до минимума. При ограниченной зоне движения простой резистнвный тензометр в достаточной мере удовлетворяет этому требованию, хотя в некоторых случаях возникает необходимость в использовании более сложных устройств, например резистивных или индуктивных потенциометров.
Там, где резистивные тензодатчики предназначаются для довольно точных ПОЗИЦИОННЫХ измерений, желательно применять двухконтурный тип обратной связи. Это помогает преодолевать трудности, вызываемые, например, температурными изменениями и изменениями сопротивления питающей сети. Важно также предусмотреть канал, обладающий хорошей тепловой проводимостью для ограничения повышения температуры за счет рассеивания энергии в резистивном элементе тензодатчика. Необходимо обеспечить хорошую защиту резистивных тензодатчиков от проникновения влаги, так как вызываемое ею изменение сопротивления изоляции может оказывать исключительно большое воздействие на датчик.
Резистивные тензодатчики обычного типа предназначены в основном для незначительных удлинений, и поэтому там, где необходимо контролировать движение большою размаха, приходится устанавливать рычаги. Одной из возможностей, реализуемых в СИИ, является использование тензодатчиков, изготовленных из электропроводящей резины или пластмассы. Они имеют преимущество, состоящее в том, что они обеспечивают измерение и управление при очень больших удлинениях.
Применяя резистивные тензодагчики, можно предусмотреть, чтобы материал основания, на которое устанавливается датчик, и материал, из которого он сделан, взаимно компенсировали температурные влияния. Другими словами, механический температурный коэффициент расширения материала основания используется для компенсации электрического температурного коэффициента сопротивления материала датчика. Имеются тензодатчики с соответствующими температурными коэффициентами сопротивления,

способные компенсировать тепловое расширение большинства распространенных металлов.
Одним из недостатков резистивных тензодатчиков является наличие предела усталости у используемого материала.
В некоторых случаях представляется привлекательным применение пьезоэлементов.
Существует множество других методов определения деформаци, например емкостный или индуктивный.
Используются тензодатчнки, резистивный элемент которых выполнен из полупроводникового материала, например кремния. Эти тензодатчики способны обеспечить выходное напряжение, почти в 100 раз большее, чем обычные тензодатчики.
Пьезоэлектрические тензодатчики используются в качестве датчиков осязания. Пьезоэлектрические элементы могут использоваться также в маломощном приводе для преодоления трудностей, связанных с гистерезисом, ползучестью и нелинейностью; при этом высокие уровни сигналов компенсируются за счет обратной связи от кремниевых тензодатчиков, приклеенных к кристаллу.
При необходимости в использовании большого количества тензодатчиков, расположенных близко друг к другу, целесообразно применять импульсное возбуждение, поскольку оно позволяет не только тысячекратно увеличивать уровни сигнала, но и своевременно разделять сигналы от разных датчиков, что сводит к минимуму взаимные помехи, возникающие из-за близости расположения.

2. Системы зрения СИИ

Системы зрения СИИ обычно снабжаются несколькими ТВ-камерами, способными оцифровывать получаемую информацию и обрабатывать ее. Центральный процессор анализирует эту информацию и распознает объект, визуальную ситуацию или сцену Основные приложения систем технического зрения относятся к управлению и классификации объектов
Типичная система технического зрения состоит из ТВ-камеры, связанной с компьютером через буферную память и препроцессор видеоинформации. Буферная память играет роль видеобуфера. Видеосигнал может направляться на ТВ-монитор для непрерывного отображения анализируемой сцены. Обработка зрительной информации о сцене в общем случае происходит очень медленно, и поэтому обычно используется препроцессор для того, чтобы сохранять

только полезную информацию и сокращать время обработки. Образы должны быть контрастными относительно фона, и для установки интенсивностей различных частей сцены в белый или черный тон используется некоторый порог. На рис. 1 показана общая схема типичной системы технического зрения СИИ.
ТВ-камеры могут быть расположены как на манипуляторе СИИ так и отдельно. Как было показано в примере 8.2 2, движение робота определяется разностным сигналом ошибки между расположением объекта и расположением камеры. Недостатком этой системы является то, что требуется наличие прочной камеры, потому что она подвергается грубым перемещениям. Кроме того, дополнительный вес камеры нежелателен и требует дополнительных алгоритмов управления и времени вычисления управления. С другой стороны, можно расположить зрительную систему вне системы ИИ (1).

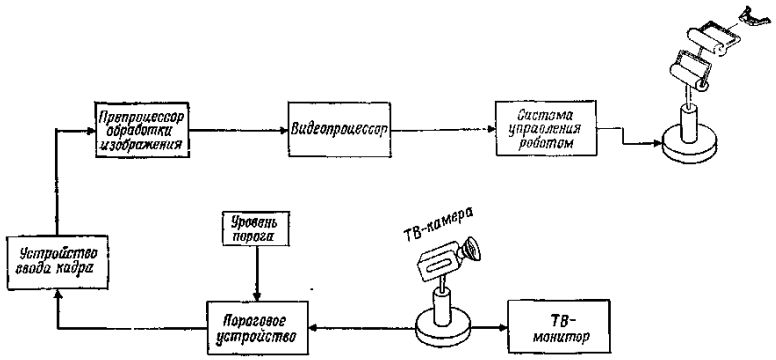


Рис. 1. Структура использования технического зрения в СИИ.

2.1. ТВ-камеры — глаза систем ИИ

Датчики систем технического зрения в основном принадлежат к типу ТВ-камер. В этих камерах, которые играют роль глаз СИИ, образы передаются в реальном времени и уже в цифровом виде. Электроника камеры способна осуществлять развертку целого изображения на более чем 500 строк (525 строк в США), что называется *кадром.* Каждый кадр изображения передается двумя полукадрами, каждый из которых образован разными строками развертки в кадре. Электроника камеры генерирует и передает полукадры последовательно. Например, передаваемое изображение состоит из выводимых на экран нечетных

строк развертки, чередующихся с четными строками. В настоящее время существует много различных типов передающих камер роботов, используемых для получения ТВ-изображений сцен. Рассмотрим некоторые из них

Ортиконовая трубка. Это трубка запоминающего типа, поскольку она основана на нейтрализации положительных зарядов сканирующим электронным лучом. Ортиконовая трубка состоит из зрительной и считывающей секций. В зрительной секции свет сцены фокусируется на полупрозрачный фотокатод. Фотокатод эмиттирует электроны, которые фокусируются магнитным полем катушки и далее ускоряются по направлению к положительно заряженной мишени. Эта мишень сделана из тонкого стеклянного диска с тонкопроволочным сетчатым экраном со стороны катода. Когда электроны ударяются об экран, со стеклянного диска эмиттируются вторичные электроны и сторона развертки стеклянного диска становится положительно заряженной. Структура этих положительно заряженных областей соответствует структуре интенсивности света изучаемой сцены. Обратная сторона стеклянного диска затем сканируется электронным лучом из электронной пушки (рис. 2) на задней стенке трубки.

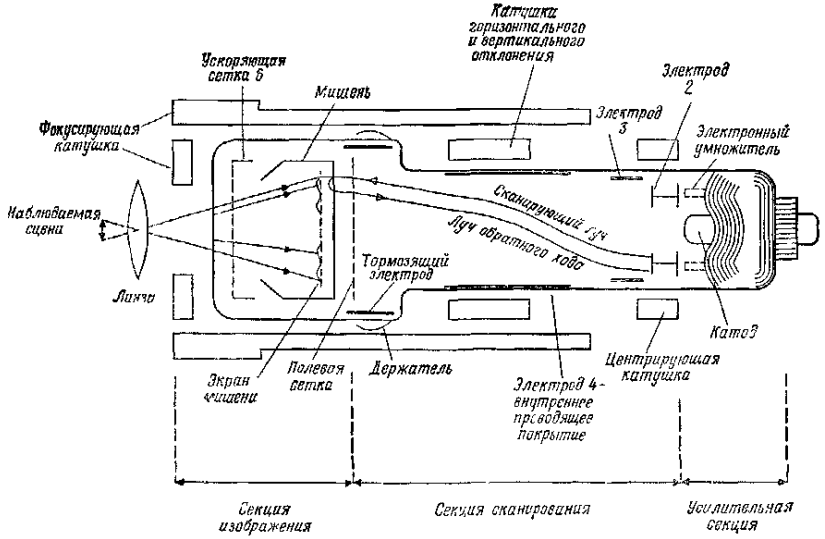


Рис. 2. Ортиконовая трубка.

Окончательное изображение представляется лучом с амплитудно-модулированной интенсивностью.

Видиконовая трубка. Эта трубка в общем случае гораздо прочнее, меньше, легче и портативнее по сравнению с ортиконовой трубкой. Мишень в видиконовой трубке покрыта прозрачной проводящей пленкой, которая действует как электрод для видеосигналов. Пленка находится на тонком фоточувствительном слое, состоящем из большого числа крошечных резистивных элементов. Эти элементы таковы, что их сопротивление уменьшается при освещении. Именно этот слой сканируется растровым способом электронным лучом из электронной пушки на задней стенке трубки. Электроны попадают на этот слой и уменьшают его потенциал. По существу две поверхности мишени образуют конденсатор, а сканирование луча порождает емкостный ток в видеосигнальном электроде, который в действительности является видеосигналом (рис. 3).

-

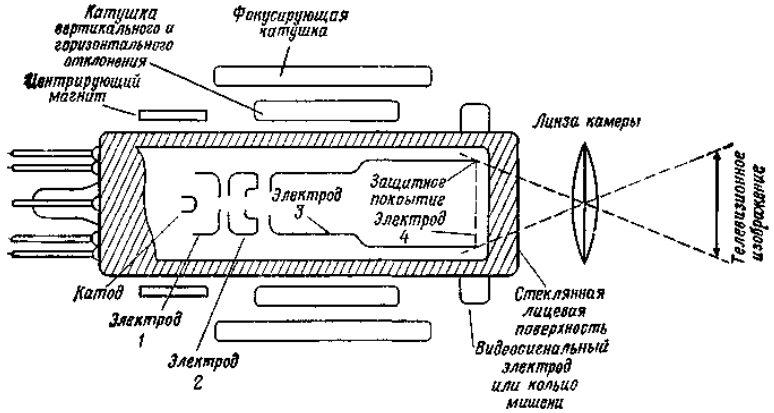


Рис. 3. Видиконовая трубка.

Плюмбиконовая трубка. Эта трубка в основном такая же, как и видиконовая, за исключением того, что фоточувствительный слой сделан из окисла свинца. Это дает некоторые преимущества перед видиконовой трубкой: малый ток, более высокая чувствительность и малая задержка.

Иконоскоп. В иконоскопе мишень сканируется электронным лучом. Мишень состоит из тонкой слюдяной мозаичной пластинки, которая покрыта фоточувствительным слоем. В иконоскопе электронный и световой лучи попадают на одну и ту же сторону поверхности мишени в отчичие от ортиконовой, видиконовой и плюмбиконовой трубок. Обратная сторона тонкой слюдяной пластинки покрыта проводящей

пленкой, которая связана с выходной нагрузкой. Эта электронная система похожа на матрицу из маленьких конденсаторов, которые разряжаются через общую нагрузку.

Диссекторная видеотрубка. В диссекторной вндеотрубке свет с исследуемой сцены фокусируется на катоде, покрытом фото-чувствительным слоем. В таком случае катод испускает электроны, количество которых пропорционально количеству света, падающего на катод. Эти электроны ускоряются к мишени с помощью анода. Небольшое отверстие перед мишенью позволяет только небольшой части эмиттируемых катодом электронов достичь мишени. Следовательно, в некотором смысле это электронный умножитель, и весь прибор работает за счет мгновенного сканирования, а не за счет вторичной нейтрализации положительные зарядов (рис. 4).

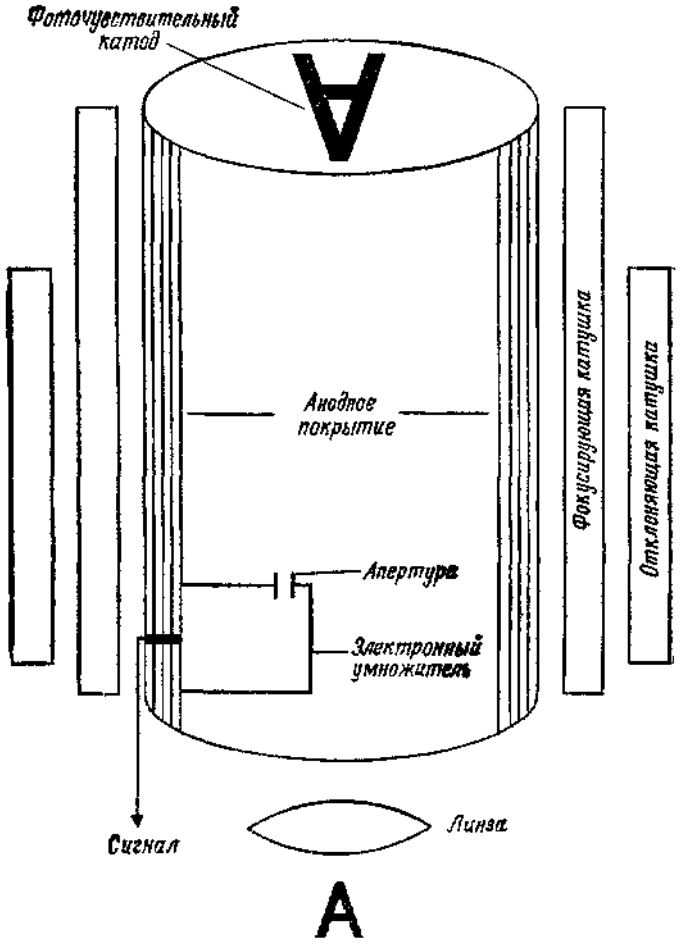


Рис. 4. Изображающая диссекторная трубка.

Для фокусировки электронного изображения и получения осевого магнитного поля используется фокусирующая катушка. Электронное изображение сканируется после прохождения отверстия мишени с помощью отклоняющих катушек. Электронный умножитель создает переменное напряжение, которое представляет собой видеосигнал. Следовательно, изображение «рассекается» при сканировании мишени.

Прибор с зарядовой связью (ПЗС). Камеры типа ПЗС попадают в общую категорию приборов с передачей заряда (ППЗ). ПЗС-камера работает примерно так же, как и полевой транзистор типа

металл — окисел — полупроводник, потому что ПЗС-камера имеет область истока и область стока, соединенные каналом из обедненной области (рис. 5).

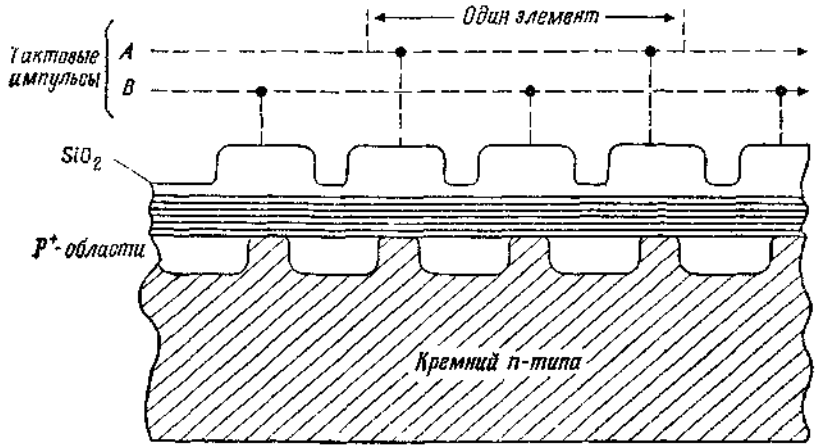


Рис. 5. Прибор с зарядовой связью (ПЗС).

По способу обработки изображения ПЗС-камеру можно рассматривать как монолитную матрицу из близко расположенных емкостей типа металл — окисел — полупроводник (МОП), которые формируют сдвиговый регистр. На ряд электродов между истоком и стоком подается серия тактирующих импульсов. Это заставляет заряды в обедненной области двигаться к выходу. Попадающие на МОП фотоны порождают заряды ПЗС-матрицы (рис. 6).

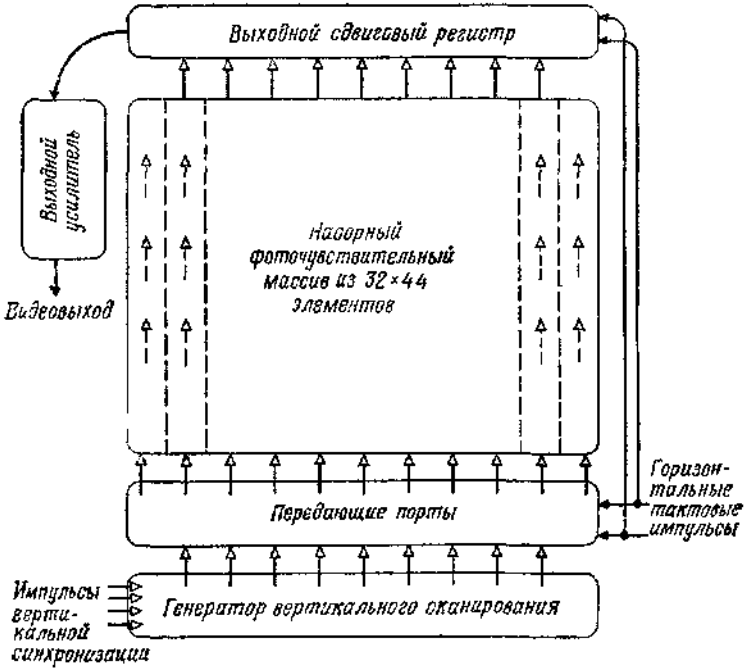


Рис. 6. Линейный передающий массив ПЗС.

Порожденные заряды переносятся к выходному регистру одновременно либо одной строкой, либо массивом Это называется соответственно *передачей строки и передачей кадра.*

Приборы с зарядовой инжекцией (ПЗИ). Камеры типа ПЗИ также попадают в общую категорию приборов с передачей заряда (ППЗ). ПЗИ в общих чертах напоминают ПЗС, за тем исключением, что во время получения изображения заряд удерживается на том месте изображения, где он был образован. Заряды считываются с помощью метода, похожего на метод *(X—Y)* адресации, используемый в памяти ЭВМ. По существу хранимый заряд инжектируется в подложку, что приводит к току смещения, который затем воспринимается как видеосигнал (рис. 7).

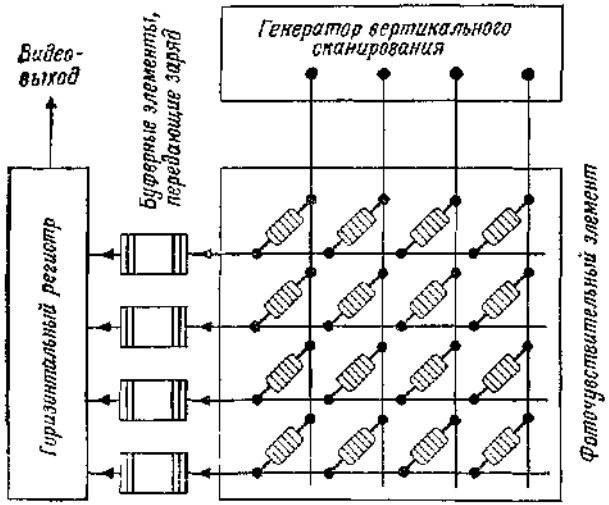


Рис. 7, Массив устройств с зарядовой инжекцией,

ПЗИ обладают рядом преимуществ по сравнению с ПЗС, а именно у них меньше дефектов, проще механизация, меньшие потери при передаче зарядов и минимальная засветка.

2.2. Основы обработки изображений в СИИ

При обработке изображения представление изображения в СИИ осуществляется в виде аналогового электрического сигнала. Исходя из этого, на первом этапе нужно оцифровать видеоинформацию, которая обычно имеет форму аналогового напряженяя (рис. 8).

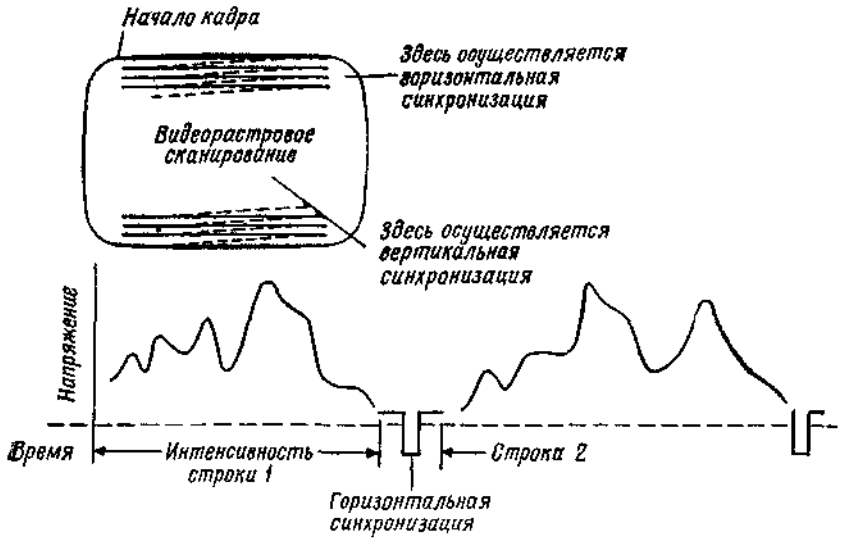


Рис. 8. Стандартный аналоговый видеосигнал. Каждая строка растрового сканирования пропорциональна яркости вдоль строки.

Здесь функцией аналого-цифрового (А/Ц) преобразователя является преобразование входного напряжения, а именно видеосигнала, и получение двоичного выходного сигнала, который пригоден для считывания цифровой ЭВМ через соответствующий интерфейс (рис.9).

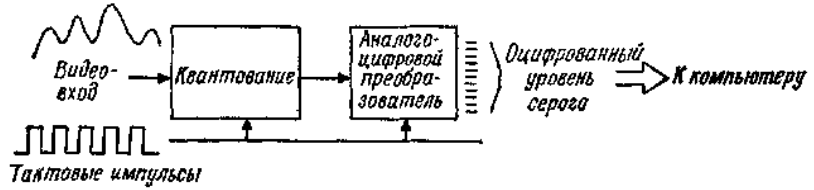


Рис. 9. Преобразование из аналоюаою видеосигнала в оцифрованный многоуровневый сигнал.

Качество А/Ц-преобразователя оценивается через его временное разрешение или скорость, с которой он может осуществлять соответствующие преобразования, а также точность его цифрового выходного сигнала.
Возможно оцифровать растровое изображение (кадр), которое состоит почти из 525 строк, с помощью 300 тактов вдоль каждой строки за

один кадровый интервал, т. е. за 1/30 с. Такая оцифровка изображений возможна с помощью существующих систем. Они осуществляют запись информации в буферную память и последующую *обработку* цифровых данных *со скоростью видеосигнала*
Оцифрованное изображение кадра хранится в памяти видеопроцессора и обрабатывается последовательно. Компьютер может тогда определять типы объектов и их ориентацию и передавать эту информацию в ЭВМ СИИ.
Существует две формы, в которых такая информация может обрабатываться: бинарное изображение и полутоновое изображение. Оцифрованное изображение состоит из элементов изображения, или элементов картинки. Яркость каждого элемента представляется числом. В бинарных видеосистемах яркость представляется 0 для фона и 1 для силуэта объекта либо 0 для черного и 1 для белого. В полутоновой системе затемненность изображения может представляться множеством значений. Например, если используются 8-битовые контроллеры, то каждый элемент изображения может иметь значение уровня яркости от 0 (черный) до 255 (белый). Системы с бинарной обработкой изображения дешевле, но они налагают ограничения на изучаемую сцену. Например, глубина объектов в общем случае не может быть измерена с использованием бинарной обработки изображения, тогда как градационная обработка позволяет производите обнаружение криволинейных поверхностей, глубины и наличие перекрывающихся частей. Градационные системы требуют больше аппаратных и программных затрат и вычислительных мощностей.
Основные этапы визуальной обработки изображений с помощью ЭВМ приведены на рис. 10.

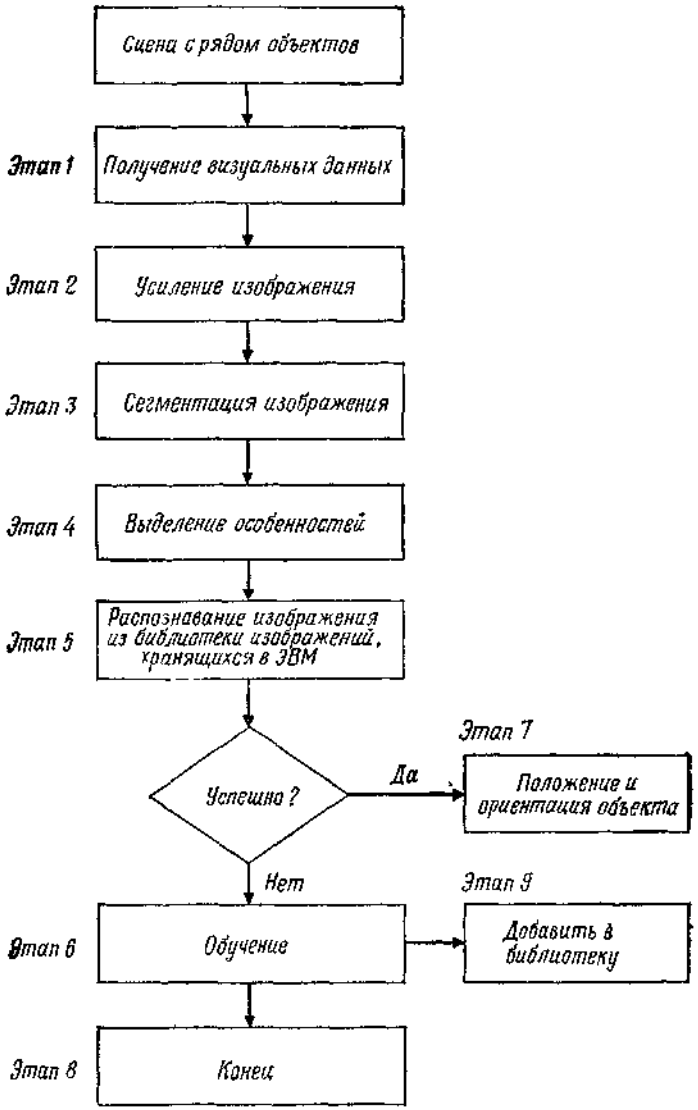


Рис. 10. Основные этапы визуальной обработки изображений.

Далее мы опишем отмеченные здесь основные этапы визуальной обработки данных.

2.3. Зрительное восприятие данных

Под зрительным восприятием данных подразумевается чтение либо горизонтальных строк развертки сверху донизу и от края до края либо выходного сигнала твердотельной ПЗС-матрицы. В любом случае необработанное изображение принимает форму двумерного массива уровней яркости, причем каждый элемент массива, соответствующий элементу изображения (элементу картинки), имеет вид, показанный на рис. 11.

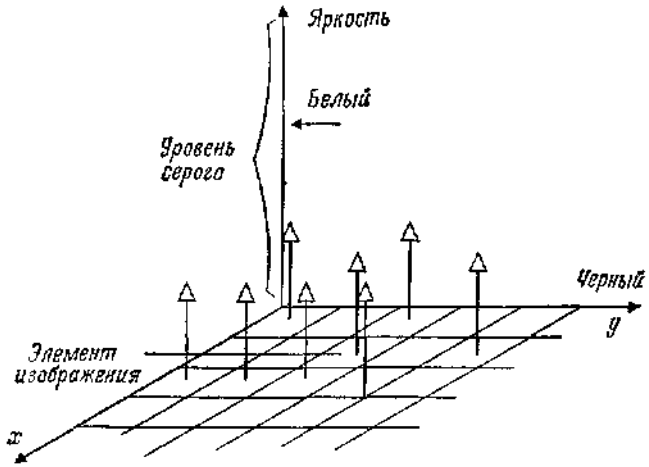


Рис. 11. Оцифровка изображения.

Здесь среднее значение интенсивности света каждого элемента изображения —это уровень серого, связанный с этим отдельным элементом. Очевидно, что **размер элемента является мерой**, **разрешающей способности рассматриваемой ТВ-камеры**.
Зрительное восприятие данных используется для решения двух задач: (1) измерение пространственных параметров объектов, которые нужно изучить визуально и (2) распознавание рассматриваемых объектов.
Операции обработки сигнала основаны на усредняющем (интегральном) методе, применяемом к видеосигналу. Однако для создания контраста и распознавания контуров изображения также необходимы дифференциальные операции.
Путем применения интегральной операции к данным может быть получено много полезной информации. Некоторыми важными характеристиками являются:
1) величина;

2) центр площади (тяжести);
3) наклон (ориентация);
4) округлость, тонкость и т. д.;
5) подобие (сходство).

Для описания этих характеристик рассмотрим рис. 12. Для случая обработки бинарных изображений введем функцию

$$I(x,\ y) = \begin{cases} 1 \text{ для всех } x,\ y \in C, \\ 0 \text{ для всех } x,\ y \notin C. \end{cases} \quad (1)$$

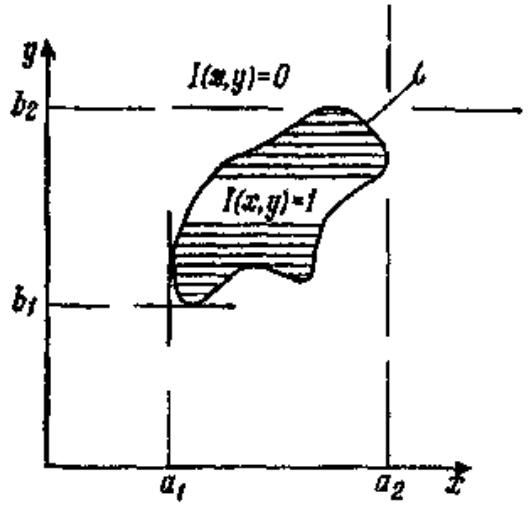


Рис. 12. Двоичная функция интенсивности *I(x, y)*.

Это функциональное определение показывает, что фигура имеет везде одинаковую яркость и может быть легко отличима от фона. Если проинтегрировать функцию *I(x,y)* по всему полю зрения, то получится величина *A,* которая представляет площадь яркой области плоского изображения

$$A = \int_{a_1}^{a_2}\int_{b_1}^{b_2} I(x.y)dxdy \quad (2)$$

Чтобы получить центр площади (или центр тяжести) изображения, предлагаются два следующих интегральных представления.

Центр площади (x_A, y_A). Он представляется как

$$\int_{a_1}^{a_2}\int_{b_1}^{b_2} sgn(x - x_A)\, I(x,y)\, dxdy = 0 \quad (3)$$

$$\int_{a_1}^{a_2}\int_{b_1}^{b_2} sgn(y - y_A)\, I(x,y)\, dxdy = 0 \quad (4)$$

Центр тяжести (x_g, y_g). Он представляется как

$$x_g = A^{-1}\int_{a_1}^{a_2}\int_{b_1}^{b_2} xI(x,y)\, dxdy \quad (5)$$

$$y_g = A^{-1}\int_{a_1}^{a_2}\int_{b_1}^{b_2} yI(x,y)\, dxdy \quad (6)$$

Для того чтобы определить ориентацию или наклон объекта, рассмотрим произвольное направление в теле, представленное линией *L* на рис 13.

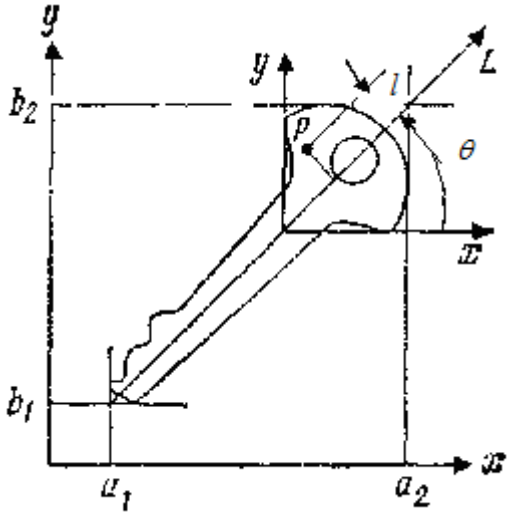


Рис. 13. Ориентация бинарного изображения.

Интегрируя квадрат расстояния l от точки *P(x,y),* принадлежащей изображению, до прямой линии, которая проходит через центр тяжести

бинарного изображения под углом θ к оси *x,* можно получить число M_2 относительно линии *L,* так что

$$M_2 = A^{-1} \int_{a_1}^{a_2} \int_{b_1}^{b_2} \frac{[(y - y_g) - (x - x_g) \operatorname{tg} \theta]^2 I(x, y)}{1 + \operatorname{tg}^2 \theta} dxdy. \quad (7)$$

Величина θ , которая обеспечивает минимум M_2, определяется как наклон бинарного изображения относительно оси *x.*

Для получения информации относитетьно других характеристик бинарного изображения, таких, как округлость и тонкость, можно использовать моменты более высокого порядка.

$$\bar{x}^2 = A^{-1} \int_{a_1}^{a_2} \int_{b_1}^{b_2} (x - x_g)^2 I(x, y) dxdy, \quad (8)$$

$$\bar{y}^2 = A^{-1} \int_{a_1}^{a_2} \int_{b}^{b_2} (y - y_g)^2 I(x, y) dxdy, \quad (9)$$

$$\overline{xy} = A^{-1} \int_{a_1}^{a_2} \int_{b_1}^{b_2} (x - x_g)(y - y_g) I(x, y) dxdy \quad (10)$$

Это та дополнительная информация, которую можно ввести в память видеопроцессора. Сходство между двумя бинарными изображениями достигается процедурой, называемой *подгонкой по шаблону,* которая рассматривается ниже. По существу различная информация, получаемая относительно бинарного изображения, сравнивается с такой же информацией относительно другого бинарного изображения, и видеопроцессор выдает различия.

Пример

Требуется вывести уравнение (7) и графически определить величину θ , которая обеспечивает минимум M_2.

Решение. Заметим, что

$$M_2 = A^{-1} \int_{a_1}^{a_2} \int_{b_1}^{b} l^2 I(x, y) dxdy. \quad (11)$$

Из рис. 14 видно, что

$$a^* = (y - y_g) - b^*, \quad (12)$$

$$b^* = (x - x_g) \operatorname{tg} \theta, \quad (13)$$

$$l = a^* \cos \theta = a^* \left(\frac{1}{1 + \operatorname{tg}^2 \theta} \right)^{1/2}. \quad (14)$$

Таким образом,

$$l^2 = \frac{[(y - y_g) - (x - x_y) \operatorname{tg} \theta]^2}{1 + \operatorname{tg}^2 \theta}, \quad (15)$$

$$M_2 = A^{-1} \int_{a_1}^{a_2} \int_{b_1}^{b_2} \frac{[(y - y_g) - (x - x_g) \operatorname{tg} \theta]^2}{1 + \operatorname{tg}^2 \theta} I(x, y) dxdy. \quad (16)$$

На рис. 14 показано решение задачи нахождения минимума.

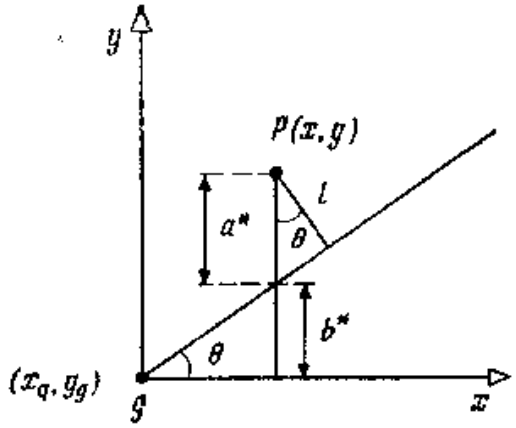


Рис. 14.

Основываясь на знании *x, y* и *xy,* можно определить следующие меры *округлости* и *тонкости:*

Округлость R : R $= A^{-1} (\bar{x}^2 + \bar{y}^2)$. (17)

Тонкость S : $S = \frac{[(\bar{y}^2 - \bar{x}^2)^2 + 4(\bar{xy})^2]^{1/2}}{\bar{x}^2 + \bar{y}^2}$ (18)

2.4. Усиление изображения

На этой стадии обработки данных обрабатывается первичное бинарное, или полутоновое, изображение для получения вторичного изображения существенно улучшенного качества и разрешения за счет удаления явного шума или других побочных явлений. В основном видеопрепроцессор используется для улучшения отношения сигнал/шум изображения, что в свою очередь уменьшает количество информации, посылаемой ЭВМ, и улучшает качество анализируемой картинки. Одной из наиболее важных черт препроцессора является то, что он четко выделяет края изображения за счет усиления разности в световой интенсивности, которая наблюдается на краю изображения, и фона. Этот процесс иногда называется *выделением края.*

2.5. Сегментация изображения

Следующий важный шаг в обработке изображений заключается в том, чтобы локализовать области изображения, которые соответствуют конкретным геометрическим объектам. На рис. 15 общее число точек на единицу площади может рассматриваться как яркость сцены изображения.

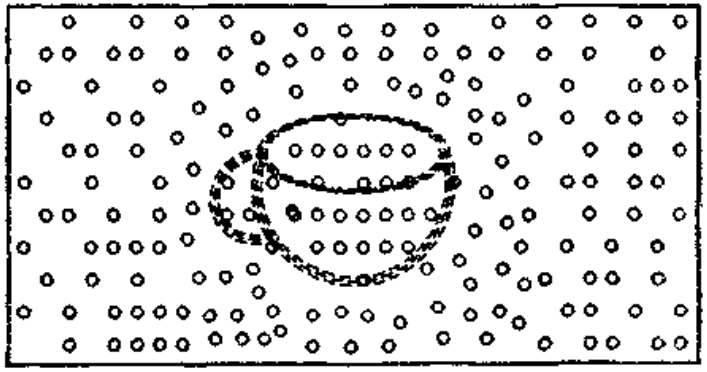


Рис. 15. Сегментация изображения

Этот процесс отделения объектов от фона называется *сегментацией изображения.* Ясно, что тщательно продуманное освещение в промышленной среде может существенно упростить процесс сегментации изображения и, следовательно, увеличить его надежность. Например, объекты могут размещаться на полупрозрачных столах с рассеянной подсветкой для получения острых контрастов. В случае прямого освещения объект может располагаться таким образом, что получается минимум теней в изображении.
Использование задней подсветки, которая резко контрастирует с передней подсветкой объектов, также улучшает контраст изображения. В процессе сегментации используется полутоновое изображение для получения бинарного изображения. Например, логический 0 может представлять объект, тогда как логическая 1 может представлять фон. Наиболее широко используемый метод сегментации полутонового изображения — это метод *пороговой сегментации изображения.* Вообще для выбора оптимальной величины порога, который отделяет объект от фона, используется гистограмма интенсивности. Например, с помощью светонепроницаемого диска, помещенного на светлый стол, можно получить гистограмму интенсивности, показанную на рис. 16, с использованием которой порог может быть выбран способом, похожим

на разделение учащихся по потокам исходя из успеваемости на основе гистограммы распределения оценок учащихся.

Рис. 16 Пороговая сегментация изображения.

2.6. Выделение и распознавиние изображения

После завершения процесса сегментации бинарного изображения образ, который оно представляет, должен быть классифицирован. Наиболее широко распространенным и **прямым методом классификации образов является сравнение образа с хранимыми моделями известных образов для нахождения подходящего.** Такие модели известны как *шаблоны.* Здесь мы определим шаблон как геометрический объект или образ, полученный или созданный путем «обучения» зрительной системы с использованием типичного представителя класса объектов. Рассмотрим объект в форме гаечного ключа и другую пару объектов, таких, как монета и штифт. Расположим эти три предмета в центре поля изображения. Теперь сегментированные изображения этих предметов можно занести в *предметную библиотеку* как представляющие шаблоны таких классов объектов. Когда неизвестный объект помещается в поле зрения, мы получаем его сегментированное изображение в виде функции $I(i, j)$, представляющей собой интенсивность элементов изображения в точках (i, j). Затем k-й хранимый шаблон $T_k(i,j)$ извлекается из библиотеки и накладывается на изображение до тех пор, пока не будет получено наилучшее соответствие. Если $T_k(i - x, j - y)$ представляет собой шаблон $T_k(i,j)$, сдвинутый на (x,y), то задача заключается в том, чтобы найти (x, y), минимизирующие ошибку $E_k(x,y)$, где по определению имеем

$$E_k(x, y) = \sum_i \sum_j [I(i, j) - T_k(i - x, j - y)]^2, \quad k = 1, 2, \ldots, n - 1. \tag{19}$$

Здесь суммирование по i и j осуществляется для перекрывающихся частей изображений $I(i,j)$ и $T_k(i,j)$ Минимальное значение $E_k(x,y)$ вычисляется для k от 1 до n, где n — число шаблонов в библиотеке. Например, предположим, что найдены $\hat{x}, \hat{y}$ и I, такие, что

$$E_l(\hat{x}, \hat{y}) < E_k(x, y) \qquad (20)$$

для всех k в библиотеке и для всех x и y в поле зрения. Если ошибка $E_l(\hat{x}, \hat{y})$ достаточно мала, т. е. $E_l(\hat{x}, \hat{y}) \approx 0$, то это отличное совпадение, и изображение $I(i,j)$ соответствует предмету Кроме того, объект $I(i.j)$ сдвинут на $(\hat{x}, \hat{y})$ относительно шаблона k. Если оптимальная ошибка $E_l(\hat{x}, \hat{y})$ слишком велика, то соответствия не будет обнаружено. Следовательно, при таких обстоятельствах предмет, породивший изображение $I(i,j)$, будет классифицироваться как неизвестный

Лучшая мера, известная как *нормированная взаимно-корреляционная* мера, определяется как

$$m_k(x, y) = \frac{\sum_i \sum_j I(i, j)\, T_k(i-x, j-y)}{\left(\left(\sum_i \sum_j I^2(i, j)\right)\left(\sum_i \sum_j T_k^2(i-x, j-y)\right)\right)^{1/2}}. \qquad (21)$$

Эта мера нечувствительна к абсолютной интенсивности и контрасту, и если она достигает максимального значения, равного 1, то это указывает на точное соответствие.

Иногда удобно представлять шаблоны набором подшаблонов с некоторыми специфичными пространственными соотношениями. Тогда становится легче сравнивать подшаблоны. Метод сравнения по шаблону в соответствии с формулой (19) инвариантен относительно сдвига предмета, пока предмет целиком находится в поле зрения. Однако мера (19) не является инвариантной относительно вращения, масштабирования или перспективных преобразований. Эти дополнительные преобразования можно было бы включить в процедуру сравнения по шаблону простым применением их либо к шаблонам, либо к изображению в видеопроцессоре, поскольку мы имеем дело только с двумерными изображениями.

Типовая камера состоит из линз и плоскости, на которой формируется изображение. Как правило, процесс получения изображения таков, как на рис. 17, где показаны оптический центр C, плоскость изображения I и расстояние f, которое представляет собой фокусное расстояние изображающей системы.

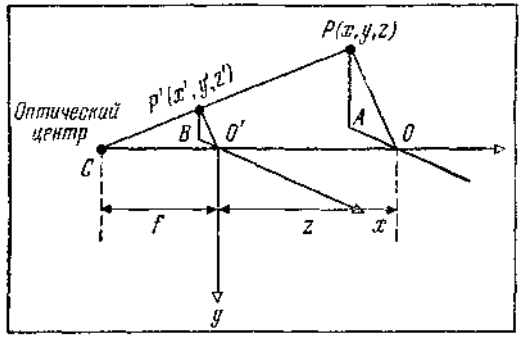


Рис. 17. Плоская сцена, полученная при помощи ТВ камеры.

Пример

Требуется показать, что точки x', y' и z' определяются следующим образом:

$$x' = \frac{fx}{f+z}, \quad y' = \frac{fy}{f+z}, \quad z' = 0. \qquad (22)$$

Решение Из подобия треугольников PAO и $P'BO'$ следует, что

$$\frac{P'B}{PA} = \frac{x'}{x} = \frac{P'O'}{PO} = \frac{BO'}{AO} = \frac{y'}{y}. \qquad (23)$$

Из подобия треугольников $P'O'C$ и POC следует, что

$$\frac{P'O'}{PO} = \frac{f}{f+z}, \qquad (24)$$

и тогда из формул (23) и (24) можно вывести соотношения (22). Преобразования (22) называются *перспективными* и могут выполняться видеопроцессором. Однако они нелинейны и их нужно линеаризовать для алгебраических операций матричного типа. Эти преобразования могут быть линеаризованы с помощью однородных координат и преобразований. Рассмотрим, например, следующее однородное преобразование:

$$\begin{bmatrix} x'_h \\ y'_h \\ z'_h \\ w'_h \end{bmatrix} = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & f^{-1} & 1 \end{bmatrix} \begin{bmatrix} x_h \\ y_h \\ z_h \\ s_h \end{bmatrix}, \qquad (25)$$

где однородные координаты определены как

$$x_h w_h^{-1} = x, \quad x'_h w'^{-1}_h = x' \quad \text{и т. д.} \qquad (26)$$

Соотношения между первичными и вторичными координатами принимают вид

$$x'_h = x_n, \quad y'_h = y_h, \quad z'_h = 0, \quad w'_n = f^{-1} z_1 + w_h. \tag{27}$$

Деля (27) на w_h, получим

$$x' = w_n^{-1} x'_h = \frac{fx}{f+z}, \tag{28}$$

$$y' = y'_h w_n^{-1} = \frac{fy}{f+z}, \tag{29}$$

$$z' = 0. \tag{30}$$

Таким образом, мы получили идентичные перспективные преобразования. Приведенные выше преобразования не дают какой-либо информации о дальности объекта Преобразование T_A вида

$$T_A = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & f^{-1} & 1 \end{bmatrix} \tag{31}$$

приведет к такому z', что

$$z' = \frac{fz}{f+z} \tag{32}$$

Это может быть использовано для получения дополнительной информации об объекте, такой, как скрытые линии или тени.
Как отмечалось выше, для улучшения процесса распознавания образов могут быть использованы другие виды преобразований. Эти преобразования включают вращение $\boldsymbol{R}$, перенос **T** и масштабирование **S**. Например, вращательное преобразование имеет вид

$$\mathbf{R} = \begin{bmatrix} & R & & 0 \\ & {\scriptstyle (3\times 3)} & & 0 \\ & & & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \quad \text{и} \quad \mathbf{R}'^{T} = \mathbf{R}'^{-1}. \tag{33}$$

Преобразование сдвига **T** имеет вид

$$\mathbf{T} = \begin{bmatrix} 1 & 0 & 0 & x_0 \\ 0 & 1 & 0 & y_0 \\ 0 & 0 & 1 & z_0 \\ 0 & 0 & 0 & 1 \end{bmatrix}, \tag{34}$$

и масштабирующее преобразование **S** имеет вид

$$\mathbf{S} = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & s \end{bmatrix}. \tag{35}$$

2.7. Сравнение модели с объектом

Получив изображение объекта в библиотеке видеопроцессора, мы начинаем выбирать такую модель m, чтобы изображение, преобразованное с помощью T_m, имело минимальное отличие от наблюдаемого изображения обьекта. Следовательно, нужно выбрать преобразование T_m для минимизации ошибки e_m между множеством точек v_k изображения объекта и точек v'_k изображения модели из множества точек модели v_m Они определяются как вершины, такие, что

$$V_p = \{v_{ip},\ i = 1,\ 2,\ \ldots,\ n\}, \tag{36}$$

$$V'_p = \{v'_{ip},\ i = 1,\ 2,\ \ldots,\ n\}, \tag{37}$$

$$V_m = \{v_{im},\ i = 1,\ 2,\ \ldots,\ n\}, \tag{38}$$

где v'_k — вершины модели с n вершинами, подвергающимися преобразованию T_m Вообще, хотелось бы минимизировать ошибку e_m

$$e_m = \sum_{i=1}^{n} |v_{ip} - v'_{ip}|. \tag{39}$$

Пусть теперь точка модели u_{im} представляется однородным вектором, т. е.

$$v_{im} = [x_{im}\ \ y_{im}\ \ z_{im}\ \ w_{im}]^T. \tag{40}$$

Аналогично точка изображения объекта v_{ip} может представляться как

$$v_{ip} = [X_{ip}\ \ Y_{ip}\ \ W_{ip}]^T. \tag{41}$$

Эти две системы координат различаются в том смысле, что точки модели находятся в эталонной трехмерной системе координат, тогда как точки изображения находятся в двумерной системе координат в плоскости изображения. Пусть теперь V_k, V'_k и V_m представляют матрицы, соответствующие точкам картинки, точкам изображения и точкам модели соответственно. Таким образом, V_k и V'_k — матрицы размера $3\times n$, тогда как V_m—матрица размера $4\times n$. Предположим, что существует преобразование H размера 3×4, такое, что, будучи примененным к V_m, оно преобразует модель $\mathbf{V}_m$ в изображение $\mathbf{V}_k$,

которое далее будет корректироваться путем масштабирования каждой точки. Если $e_m = 0$, имеем

$$\mathbf{H}\mathbf{V}_m = \mathbf{V}_p\lambda, \tag{42}$$

где λ — диагональная матрица размера $n \times n$ с n неизвестными, т. е. $[\lambda_1, \lambda_2 \ . \ , \lambda_n]$.

Пример

Требуется найти решение для **H**, если $\lambda = I$, где I — единичная матрица.

Решение. Из уравнения (42) видно, что

$$\mathbf{H}\mathbf{V}_m\mathbf{V}_m^T = \mathbf{V}_p\lambda\mathbf{V}_m^T. \tag{43}$$

Теперь матрица $\mathbf{V}_m\mathbf{V}^T_m$ является матрицей размера 4×4 и может быть преобразована так, что

$$\mathbf{H}\left(\mathbf{V}_m\mathbf{V}_m^T\right)\left(\mathbf{V}_m\mathbf{V}_m^T\right)^{-1} = \mathbf{V}_p\lambda\mathbf{V}_m^T\left(\mathbf{V}_m\mathbf{V}_m^T\right)^{-1} \tag{44}$$

или

$$\mathbf{H}\mathbf{I} = \mathbf{H} = \mathbf{V}_p\mathbf{V}_m^T\left(\mathbf{V}_m\mathbf{V}_m^T\right)^{-1}. \tag{45}$$

2.8. Оценка поворота изображения

Определение угла поворота изображения объекта является важной задачей СИИ. Здесь мы определим ряд интегральных мер, а именно моментов, для оценки угла поворота изображения.

Предположим, что функция интенсивности сцены $I(x,y)$ задана для бинарного или полутонового изображения следующим образом:

$$I_b(x, y) = \begin{cases} 1 & \text{при } I(x, y) \geqslant T_b, \\ 0 & \text{в противном случае} \end{cases} \tag{46}$$

или

$$I_g(x, y) = \begin{cases} I(x, y) & \text{при } I(x, y) \geqslant T_g, \\ 0 & \text{в противном случае,} \end{cases} \tag{47}$$

где T_b и T_g — пороги (подавляющие или отделяющие фон).

Смешанные (декартовы) моменты изображения M_{mn} определяются следующим образом

$$M_{mn} = \iint x^m y^n I(x + x_g, y + y_g)\, dx\, dy, \tag{48}$$

где (x_g,y_g)—координаты центра площади изображения. Полярные моменты изображения определяются так

$$S_{mn} = \iint r^{n+1}(\cos m\theta)\, I(r\cos m\theta + y_c r \sin m\theta + y_0)\, dr\, d\theta, \tag{49}$$

$$R(i, j) = \mathbf{V}I(i, j)_P. \tag{50}$$

Вектор полярного момента $\mathbf{P}_{mn}$ определяется так

$$\mathbf{P}_{mn} = \begin{bmatrix} C_{mn} \\ S_{mn} \end{bmatrix} \tag{51}$$

и сохраняет длину: если изображение поворачивается на угол θ*, то $\mathbf{P}_{mn}$ поворачивается на $m\theta$*

2.9. Калибровка телевизионной камеры

Преобразования, которые связывают координаты объекта в абсолютной системе координат с координатами изображения, известны как *преобразования камеры.* Чаще всего камера жестко установлена в системе координат, и поэтому полезно измерить или откалибровагь параметры, которые влияют на преобразования камеры. Процедура заключается в том, чтобы рассмотреть определенный трехмерный объект в данных позиции и ориентации и измерить точки в изображении, соответствующие заданным точкам объекта. Затем можно найти необходимое преобразование, основанное на уравнениях, приведенных в разд.1.2.6

Калибровка камеры дает величины параметров модели, которые нужны для вычисления линии визирования в пространстве, соответствующей точке в плоскости изображения. При заданных трехмерных координатах точки в поле зрения преобразования камеры позволяют вычислить координаты изображения. В обратном преобразовании камеры через координаты изображения определяется линия визирования в пространстве. В СИИ используется именно обратное преобразование, когда нужно рассчитать расположение объекта на основе изображения.

Для визуального стереоанализа используются две камеры, и становится важной калибровка двух камер, выдвигающая дополнительные проблемы. В литературе (описываются два метода калибровки камер, которые главным образом основываются на модели вал — отверстие и модели, включающей несколько плоскостей.

2.10. Выделение изображения

Выделение изображения является важной частью зрения СИИ. Например, выделение линий, которые соответствуют краям объекта, особо важно по очевидным соображениям. Обычно выделение основывается на предположении, что интенсивность света постоянна либо плавно меняется на грани объекта и может терпеть разрыв только на линии пересечения граней.

В плоскости непрерывного изображения градиент функции интенсивности в общем случае конечен и задается функцией градиента Робертса *R(i,j),* такой, что

$$R(i,\ j) = \nabla I(i,\ j)_F \tag{52}$$

или

$$R(i,\ j) = \{[I(i+1,\ j+1) - I(i,\ j)]^2 + [I(i,\ j+1) - I(i+1,\ j)]^2\}^{1/2} \tag{53}$$

где *I(i,j)* — интенсивность изображения в точке элемента (*i,j*).
Направление градиента определяется через α, такое, что

$$\alpha = -\frac{\pi}{4} + \operatorname{arctg}\left(\frac{I(i,\ j+1) - I(i+1,\ j)}{I(i+1,\ j+1) - I(i,\ j)}\right). \tag{54}$$

Считается, что край оказывается в точке (*i,j*), если оператор Робертса *R(i,j)>r*,* где *r** — выбранный порог. Этот метод позволяет выделить все неоднородности на поверхности либо узнать текстуру граней изображения.
Для выделения контура и записи всей информации, полученной для внешних и внутренних контуров, можно использовать рекурсивные алгоритмы. Иногда перед выделением изображения необходимо ликвидировать погрешности, которые возникают из-за оцифровки изображений. Рассмотрим, например, кольцо и его оцифрованное бинарное изображение на рис. 18.

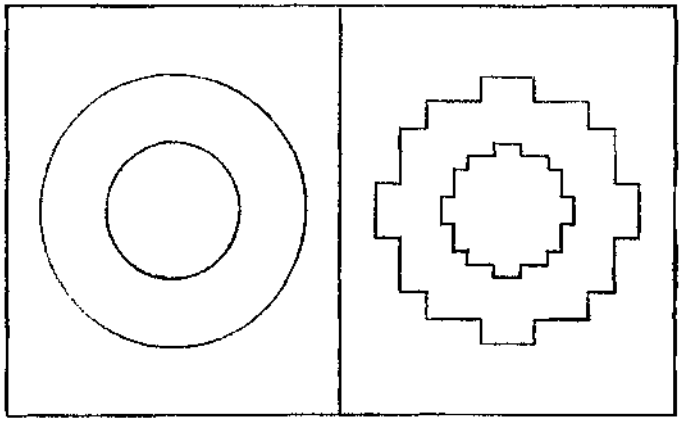


Рис. 18. Объект и его оцифрованное бинарное изображение

Можно выбрать ряд параметров для их сравнения и выделения правильной формы объекта на основе его оцифрованного изображения, а именно такие, как:
1) полная площадь,
2) моменты,

3) максимальный и минимальный радиусы,
4) расположение центра фип'ры,
5) периметр,
6) число отверстий

или любая другая измеряемая характеристика.

3. Элементы исследования в области СИИ

Целью исследований является обеспечение практического применения СИИ.
Ввиду ограниченности объема в данном разделе будет дано лишь краткое общее описание СИИ и будут вкратце рассмотрены вопросы их применения.
Блок-схема СИИ показана на рис. 1.

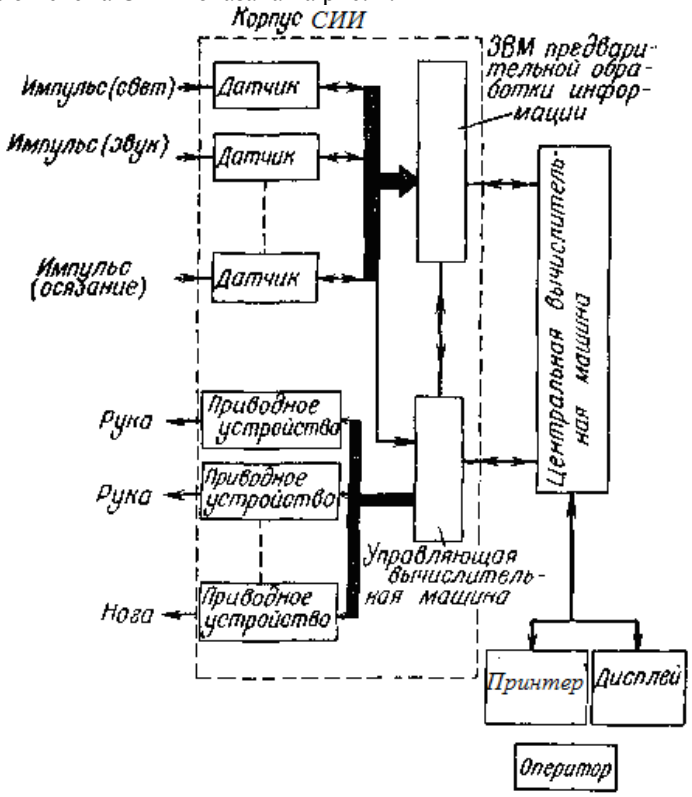


Рис. 1. Блок-схема СИИ.

Роль мозга в нем играет центральная ЭВМ, функции органов зрения выполняют телевизионные камеры и специальные ЭВМ, органов слуха — микрофоны и специальные ЭВМ.

1) Зрительная система. Зрение позволяет человеку выполнять сложные операции. Подобно этому и возможности СИИ при снабжении ее органами зрения резко возрастают. От органов зрения требуется выполнение более сложных функций, чем распознавание простых образов, подобных буквам алфавита.

При рассмотрении ряда технологических процессов обнаруживается довольно много случаев, когда можно обойтись сравнительно простой конструкцией органов зрения. В подобных случаях функции органов зрения сводятся к установлению наличия или отсутствия объекта, определению его местоположения, его классификации по предварительно установленным простым признакам и т. д. Системы ИИ для реализации этих функций содержат фотоэлементы, телекамеры, разнообразные осветительные установки и сравнительно простые электронные схемы.

Необходимые органам зрения функции определяются видом обслуживаемого технологического процесса. Однако подобным зрительным системам свойственны определенные ограничения; поэтому обойтись этими простейшими функциями глаза можно лишь при условии упорядочения обстановки, в которой выполняется операция, а также содержания самой операции.

Наиболее важной проблемой при создании СИИ является разработка глаза, способного до некоторой степени различать простые объекты. Одной из таких разработок является глаз СИИ Института комплексных исследований электронной техники. На рис.2 приведена блок-схема указанной системы.

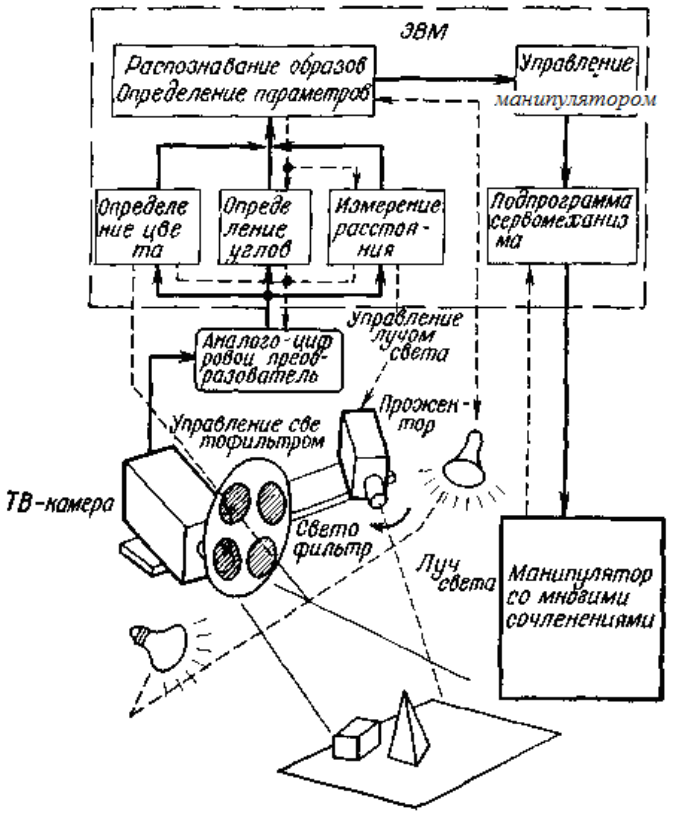


Рис. 2 Блок-схема зрительной системы ИИ

Как видно из приведенного описания, большая часть обработки информации осуществляется на ЭВМ. В рамках данного исследования определялась возможность объемного распознавания предметов при условии введения некоторых ограничений. В условиях производства не представляет труда изменить расположение объектов таким образом, чтобы глаз СИИ мог их распознать. С помощью ЭВМ направленность освещения регулируется так, чтобы и фон и объект стали монохромными. Это упрощает процесс распознавания. Что касается формы предметов, то разработчики системы старались иметь дело с простыми трехмерными предметами, ограниченными плоскостями, цилиндрическими поверхностями и т. п.

В качестве устройства ввода используются промышленные телевизионные камеры. При проведении экспериментов по распознаванию цветов свет, попадающий на вход в телевизионную камеру, предварительно пропускается через красно-зелено-синие фильтры, установленные на вращающемся диске. С целью экономии объема памяти сигналы от телевизионной камеры преобразуются в аналого-цифровом преобразователе в сигналы двух видов.
Аналогичное явление происходит и в органе зрения человека. Как показано на рис. 3, глаз имеет область с высокой разрешающей способностью в пределах ±1° от центра сетчатки, остальная же поверхность сетчатки имеет небольшую разрешающую способность.

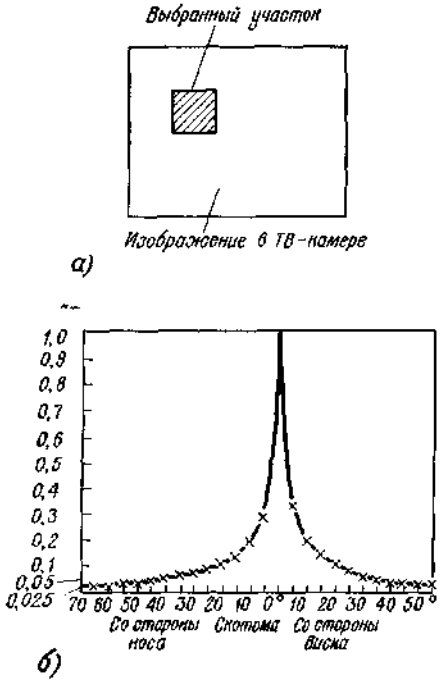


Рис. 3 Выделение зоны и распределение остроты зрения по сетчатке *а* — выделенная зона на ТВ изображении; *б* — распределение остроты зрения по сетчатке

Поле зрения телевизионной камеры делится на 64× 64 = 4096 точек, яркость каждой из которых преобразуется в цифровой сигнал, направляемый в ЭВМ. На основании полученной информации

определяется местонахождение предмета. Далее, с целью более подробного изучения формы предмета, выявленная зона делится снова на 4096 участков (заштрихованная зона на рис. *3, а).* Косыми линиями производится сканирование этой зоны с последующим аналого-цифровым преобразованием полученных сигналов. В случае когда объект больше выделенной зоны, она перемещается подобно тому, как человек скользит взглядом по предмету.

а) Вычисление линейного плана (определение проекции на плоскость в виде контурного изображения).

Первая стадия обработки информации состоит в том, что на основании видеосигналов, поступающих от телевизионной камеры, определяются положения углов предмета и составляется линейный план предмета. Исходя из того, что обычно на линии пересечения плоскостей резко меняется яркость отраженного света, рассчитывают пространственный дифференциал яркости изображения в целом.

Однако яркость на плоскости неодинакова, и во вводимой информации присутствует много помех. Поэтому в случае, когда яркости двух плоскостей близки, угол между ними трудно определить. В таком случае применяют пространственный фильтр, который представляет собой как бы эталон обычного зрительного восприятия изображения. Искомый угол находят тогда путем сравнения изображений, полученных с телекамеры и с фильтра. Пространственный фильтр 7×7, применявшийся в процессе исследований глаза СИИ Стэнфордского исследовательского института, показан на рис. 4.

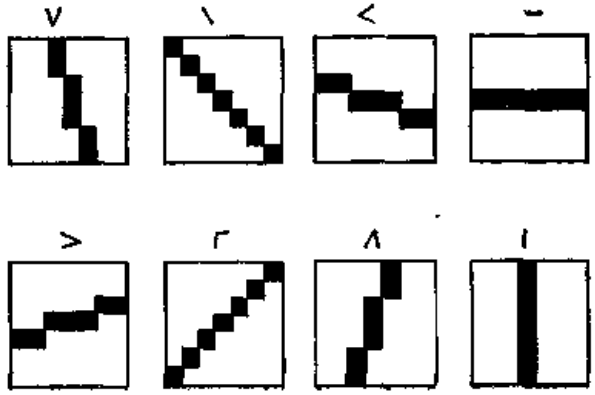


Рис.4. Пространственный фильтр, в котором в качестве модели использована клетка зрительной системы кошки (по Форсену).

Выявляются отрезки в восьми направлениях.

Подгонка коротких отрезков линий осуществляется с помощью вычислительной машины. На рис. 5 показаны контурные изображения прямоугольников, полученные в результате моделирования на ЭВМ.

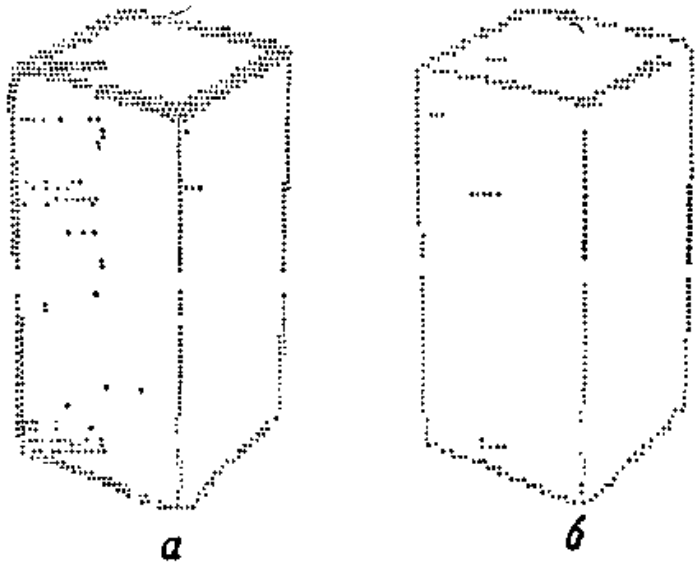


Рис. 5. Определение контуров прямоугольника (по Форсену).
а — дифференцирование пространства; *б* — изображение после фильтра, моделирующего глаз кошки

Для предметов сложной формы эффективным является метод, при котором направление освещения регулируется при помощи вычислительной машины и полученные после этого контурные изображения теоретически комбинируются. Реальный пример вычисления правильного контурного изображения, полученного из четырех незавершенных изображений, приведен на рис. 6.

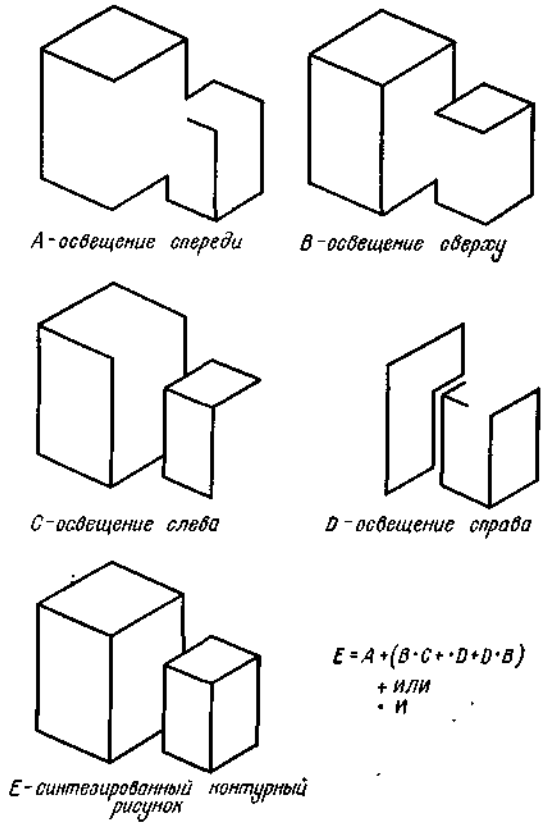


Рис. 6. Синтез контурного изображения в результате поочередного освещения предмета с четырех сторон (по Сираи).

б) Распознавание цвета. Цвет наряду с информацией о яркости и линейных размерах является важным свойством, облегчающим распознавание предмета. Восприятие цвета не зависит от угла освещения и расстояния. Используя это свойство, можно быстро найти контуры предмета.
В ЭВМ вводится видеосигнал изображения, прошедший через красно-зелено-синие светофильтры, и по нему определяют цветовые признаки каждой из точек. Поскольку проверка всех точек изображения требует слишком много времени, разработан алгоритм, сокращающий процесс обработки информации. На рис.7 показан предмет, состоящий из кубиков шести цветов.

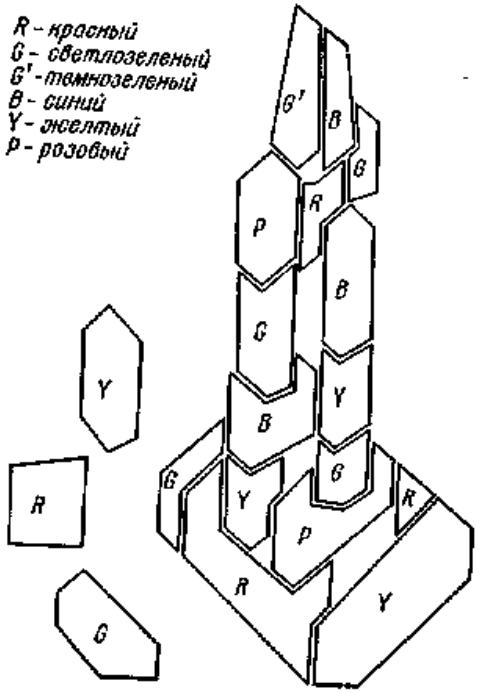


Рис.7. Пример распознаваьия цвета.

в) Измерение расстояния. Наряду с определением формы предмета СИИ должна установить его размеры и местоположение. Для измерения расстояний предлагаются следующие методы:

1) **Метод с использованием одного глаза.** Точки на плоскости, расположенные в поле зрения телекамеры, соответствуют точкам телевизионного изображения. Следовательно, если предварительно произвести настройку, можно по координатам *x, y* установить местонахождение точек на плоскости.

С использованием этого метода в СИИ по координатам *x, y* телевизионного изображения определяют координаты всех точек на горизонтальной рабочей платформе, высота которой известна. Высота предмета определяется в направлении, перпендикулярном рабочей платформе. Используемая аппаратура проста, но ей свойствен ряд ограничений при измерении расстояний. Данный метод непригоден для СИИ, работающих в полярной системе координат.

2) **Стереометод** заключается в том, что используется пара телевизионных камер и расстояние до объекта измеряется с помощью тригонометрических функций по данным о положении объекта, об угле и расстоянии от одной из камер. Сложным при этом методе оказывается выяснение вопроса о том, каким точкам одного из изображений соответствует каждая из точек другого изображения. Для того чтобы рассчитать расстояние до правого и левого изображений, Сутро устанавливал в поле зрения небольшую рамку, называемую окном, и передвигал одну из ее сторон таким образом, чтобы разница расстоянии до изображений была минимальна.

3) **Дальномер с осветительным устройством**. Расстояние до предмета можно измерить путем применения телевизионной камеры совместно с осветительным устройством. На рис. 8 показан этот принцип.

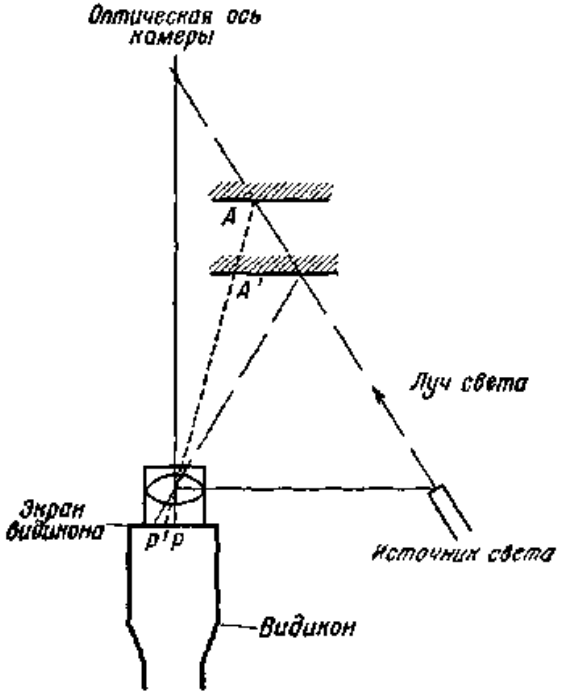


Рис. 8 Принцип измерения расстояния.

Телевизионная камера фиксирует предмет, на который проецируется пучок света, если предмет находится в точке *A,* на изображении освещается точка *P,* если в точке *A′* — точка *P′.* Следовательно, с помощью тригонометрических измерений можно определить расстояние до предмета. Для ускорения процесса измерения расстояния до каждой точки, находящейся в поле зрения, делаются попытки распознавания предметов, проецируя на них движущуюся узкую полосу света. На рис.9 в качестве примера показан предмет, очертания которого определены путем обработки на ЭВМ изображений, полученных с помощью такого метода.

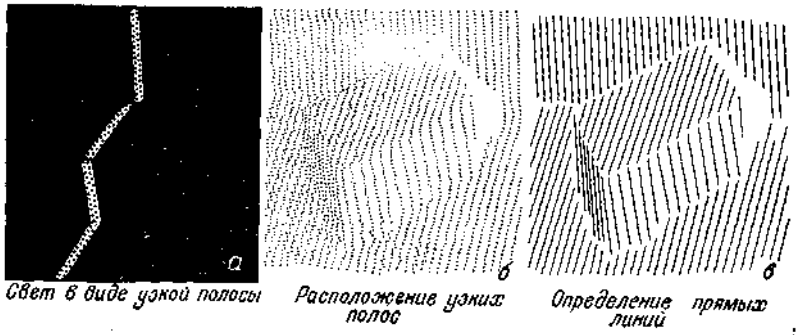


Рис. 9. Обработка информации, полученной при проецировании на предмет узких полос света
а — узкая полоса света; *б* — конфигурация полос; *в* — выявление прямых линий

На фигуре хорошо видны горизонтальные и вертикальные плоскости предмета.

г) Распознавание. Линейный план, цвет, расстояние и другие характерные признаки являются исходной информацией для программы, по которой осуществляется распознавание объектов. Существуют два основных метода распознавания объектов. При первом методе о форме объекта судят по конкретной комбинации характерных признаков, о которых сказано выше, при втором методе распознавание осуществляется сравнением объекта с прототипом.

2) Управление с помощью ЭВМ. В результате проведенных исследований разработана система управления с обратной связью от датчиков, размещенных на манипуляторе СИИ.

Информация от датчиков обрабатывается мини-ЭВМ. Создание такой системы значительно расширило возможности СИИ.

а) Управление с обратной связью от датчиков, размещенных на манипуляторе СИИ. Ранее рассматривалась система управления, в которой информация о работе в цикле находится в блоке памяти, причем эта информация была записана в блок памяти при обучении СИИ (рис.10).

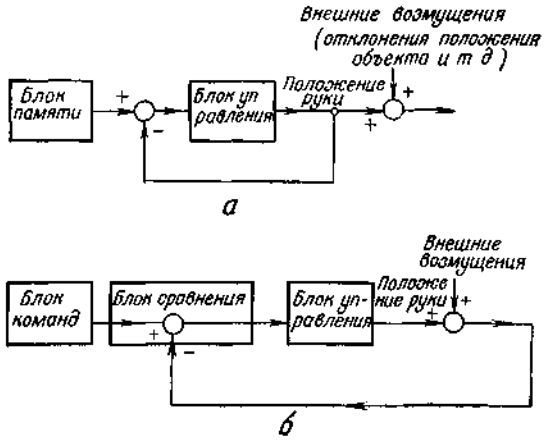


Рис.10. Способы управления рукой СИИ.
а — управление с разомкнутым контуром, б — управление с замкнутым контуром

Такая система обеспечивает высокую точность лишь при фиксированном положении СИИ и окружающих ее объектов. Всякое же изменение относительного расположения СИИ и объектов при отсутствии датчиков, вызывает погрешность.

Между управлением манипуляторо СИИ и управлением рукой человека имеются очевидные различия. Руке человека свойственны большие погрешности в отработке конкретных величин перемещения. Так, приказ «передвинуть руку на 10 см» человек выполнит с погрешностью в несколько миллиметров. Однако при манипулировании предметом рука человека имеет более высокую точность, чем манипулятор СИИ с обучением. Это объясняется приобретенными человеком навыками и наличием обратной связи от объекта.

Для осуществления управления с обратной связью от объекта необходимо снабдить СИИ элементом, чувствительным к давлению. По сигналам от такого элемента ЭВМ должна корректировать программу управления манипулятором. Использование датчиков,

придающих чувство осязания, делает возможным и ряд других способов управления СИИ. Один из таких способов приведен ниже.

б) Управление с участием человека. В Массачусетском технологическом институте Эрнст и Шеридан с сотрудниками провели исследования в области дистанционного управления СИИ. Целью исследований явилось обеспечение возможности управления системой ИИ, находящейся на Марсе, Луне или другой планете, при передаче ей минимального количества информации. Человек с Земли дает СИИ указания на простом языке, близком к естественному. Эта информация расшифровывается с помощью размещенном на СИИ компьютере и приводит СИИ в действие. У СИИ нет органов зрения и невелики возможности в части обработки информации. В непредусмотренной ситуации СИИ обращается к помощи человека на Земле и действует по его указаниям. Такой способ управления получил наименование управления с участием человека и под его наблюдением (supervisory control).

Диалог между человеком и СИИ в этой системе выполняется на алгоритмическом языке **МЭНТРАН** (MANTRAN).

Манипулятор СИИ (рис. 11) построен на базе копирующего манипулятора модели 8 фирмы AMF и имеет 7 степеней свободы.

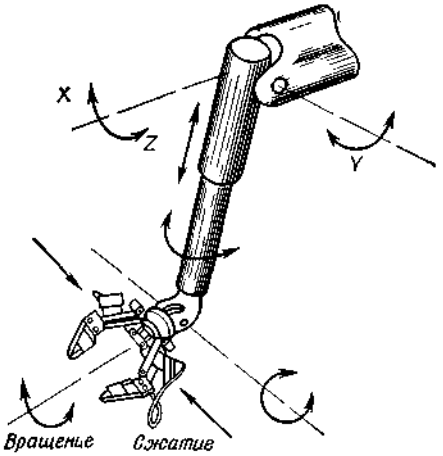


Рис.11. Многостепенный манипулятор СИИ (по Барверу).

В качестве приводных используются шаговые электродвигатели. В пальцах манипулятора СИИ смонтировано 10 датчиков давления, каждый из которых работает на свою триггерную схему. Оператор

составляет программу диалога на языке МЭНТРАН, используя простые слова, близкие к английским. Пример такой программы приведен на рис. 12.

```
Этап 1  Двигаться вперед на 1 Ø, влево на 5 ØØ до тех пор,
        пока А) коснешься слева
             Б) коснешься с других сторон
        Если условия движения выполнены — делать 2,
        если А — делать 3,
        если Б — запросить о помощи

Этап 2  Двигаться вперед на 1 Ø, вправо на 5 ØØ до тех пор,
        пока А) коснешься справа
             Б) коснешься с других сторон
        Если условия движения выполнены — делать 1,
        если А — делать 6,
        если Б — запросить о помощи

Этап 3  Двигаться назад на 4 Ø*.

Этап 4  Двигаться влево на 24, открыть на 4 ØØ*.

Этап 5  Двигаться вперед на 4 Ø, закрыть на 4 ØØ до тех пор,
        пока А) коснешься изнутри
             Б) коснешься с других сторон
        Если условия движения выполнены — запросить
                                           о помощи,
        если А — делать 8,
        если Б — запросить о помощи

Этап 6. Двигаться назад на 4 Ø*.

Этап 7. Двигаться вправо на 24, открыть на 4 ØØ до тех пор,
        пока А.
        Если условия движения выполнены — делать 5

Этап 8  Идти к месту LOC2
```

Рис.12. Пример программы MANTRAN («Ищи предмет и, если он имеется, отнеси его в LOC 2») (по Барберу).

Программой СИИ определяется задача: найти в помещении предмет и перенести его в указанное место LOC2.

Поскольку у СИИ нет органов зрения, он обыскивает помещение наощупь и обнаруживает предмет с помощью тактильных датчиков, сообщающих ему чувство осязания (рис.13).

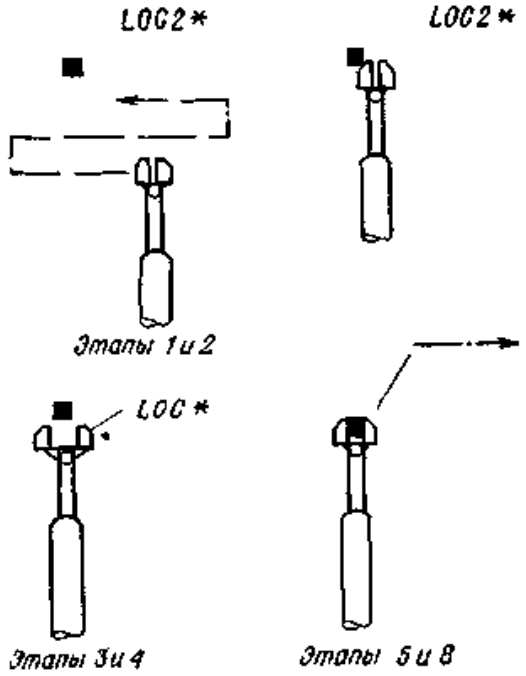


Рис.13. Движения манипулятора СИИ по программе на рис. 12 (по Барберу).

Манипулятор СИИ вначале несколько отодвигается от предмета. Затем пальцы раскрываются и, вновь продвинувшись вперед, манипулятор СИИ захватывает предмет и переносит его в указанное место. Можно усложнить программу, заставив СИИ определить форму и размеры простого предмета, взять его за определенную часть (например, за ручку) и перенести в указанное место. Следует отметить, что возможности манипулятора повышаются при использовании его в сочетании с мини-ЭВМ.

в) Сервоуправление с отражением усилия. Иноуэ из Института комплексных исследований электронной техники с помощью манипулятора, подобного показанному на рис. 11, в сочетании с мини-ЭВМ доказал возможность еще одного, более совершенного способа управления манипулятором СИИ. При этом, как показано на рис. 14,

производится предварительный расчет приложенного к манипулятору СИИ момента, исходя из отклонения каждого сочленения манипулятора СИИ от исходного положения.

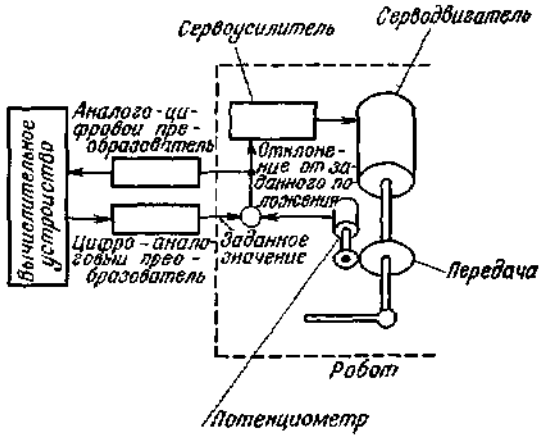


Рис.14 Сервомеханизм с отражением усилия (по Иноуэ).
Момент нагрузки пропорционален отклонению от заданного положения, что дает информацию о приложенном к манипулятору СИИ моменте

Используя эту информацию, ЭВМ осуществляет управление каждым сочленением. Благодаря этому способу управления, стало возможным выполнение следующих операций:
— различение веса захваченных предметов;
— определение того, коснулся ли захваченный предмет других предметов;
— введение штифтов в отверстия;
— вращение кривошипных механизмов.

В заключение следует отметить, что теория искусственного интеллекта еще довольно молода и переживает период бурного развития. Поэтому вполне вероятно, что системы ИИ, с которыми читателю придется столкнуться на практике, будут существенным образом отличаться от систем ИИ, описанных в данной работе.

3.1. Алгоритмы распознавания ситуаций

Задача описания, распознавания и анализа ситуаций является одной из центральных проблем построения искусственного интеллекта. Для

формализации и автоматизации решения этой задачи, как мы увидим ниже, удобно использовать **язык исчисления предикатов и адаптивную систему логического вывода.** Поскольку для системы ИИ наиболее важное значение имеет зрительная информация, мы сосредоточим внимание на методах описания, распознавания и анализа трехмерных сцен по их изображениям. Что же касается **задач переработки речевой, тактильной и другой сенсорной информации, то они могут решаться по существу теми же методами**.

Трудности, возникающие при реиении задачи распознавания отдельных предметов на сложной сцене, связаны с наличием «порочного круга»: для того чтобы распознать некоторый предмет на сцене, нужно прежде всего его «выделить», а для того чтобы «выделить» этот предмет, нужно его распознать. Это приводит к тому, чго классические методы распознавания «перцептронного» или статистического типа в этой задаче практически не применимы. На первый взгляд кажется, что выход из указанного «порочного круга» только один — полный перебор элементов изображения предмета и сцены. **Однако более лубокий анализ этой задачи позволяет ее сформулировать и решить как задачу логического вывода**.

Идея предлагаемого решения основана, во-первых, на том, что применяется предварительное обучение СИИ путем **показа ей отдельных предметов из различных классов и сообщения ей не только названия предмета, но и, возможно, его описания.** Это обучение ведется человеком в диалоговом режиме, позволяющем оперативно выявлять и исправлять ошибки СИИ. Во-вторых, специфика задачи ярко проявляется в большой вариативности изо-бражений реальных предметов и сцен, которая имеет двоякую природу. С одной стороны, она порождается естественной вариативностью характеристик самих предметов, с другой стороны, — **перемещением предметов в пространстве. Вариативность второго рода можно трактовать как результат действия некоторых известных преобразований изображения.** Априорное знание этих преобразований позволяет построить алгоритм распознавания, инвариантный по отношению к этим преобразованиям. Благодаря этому удается не только «избавиться» от вариативности второго рода, но и существенно облегчить задачу переработки зрительной информации.

Сама эта задача подразделяется на следующие **подзадачи: описание классов (формирование понятий о классах объектов), распознавание изображения данного предмета, анализ изображения сцены (распознавание всех предметов на сцене).**

Результаты решения перечисленных подзадач могут использоваться для моделирования внешней среды путем преобразования изображений предметов и сцен в их пространственное представление, а также для описания сцен на естественном языке или, наоборот, для синтеза изображения сцены по ее описанию.

Рассмотрим сначала *задачу описания классов* (**формирования понятий**). Эта задача решается в режиме обучения. Системе ИИ последовательно предъявляют предметы из различных классов (и, возможно, в различных ракурсах) с указанием, к какому классу каждый такой предмет принадлежит. Эти предметы (а также их изображения) называют **эталонными**, а совокупность классифицированных предметов — **обучающей выборкой**. По этим данным СИИ должна автоматически сформировать описание классов в терминах тех свойств предметов, которые непосредственно измеряются сенсорной системой. Примерами этих свойств, которые мы будем называть первичными признаками, являются следующие: «красное», «ближе», «правее», «выше», «две точки соединены отрезком», «два отрезка параллельны», «зоны одинаковой яркости» и т. п. Таким образом, **изображения предметов и сцен задаются полным набором своих первичных признаков**.

Каждому признаку поставим в соответствие предикат $\xi_i(x_1, \ldots, x_{ni})$, где $x_1, \ldots, x_{ni}$ — элементы изображения ω, определяющие наличие на нем i-го признака. Тогда каждому эталону ω_h (предмету из обучающей выборки) соответствует набор значений предикатов $\xi_i^{(h)}(c_1, \ldots, c_{hi})$, истинных на изображении данного предмета ω_h. Здесь c_j — предметные константы, означающие фиксированные части изображения. Описанием изображения эталона ω_h будем называть конъюнкцию $\bigwedge_{i=1}^{P_h} \xi_i^{(h)}(c_1, \ldots, c_{ni})$.

Поскольку к одному и тому же классу могут принадлежать несколько эталонов (например, предмет из этого класса, показанный в разных ракурсах), то описанием всех эталонных изображений, принадлежащих данному классу Ω_k, является дизъюнкция

$$\bigvee_{\omega_h \in \Omega_k} \bigwedge_{i=1}^{P_h} \xi_i^{(h)}(c_1, \ldots, c_{ni}).$$

Если теперь в этой дизъюнкциивсе предметные константы c_j заменить на соответствующие предметные переменные x_j, то получим ППФ, которую естественно назвать описанием класса Ω_k. Введем предикат $\sigma(k)$, означающий принадлежность изображения классу Ω_k. Тогда каждый класс Ω_k описывается аксиомой класса (АК) вида

$$\bigvee_{\omega_k \in \Omega_k} \bigwedge_{i=1}^{P_k} \xi_i^{(k)}(x_1, \ldots, x_{ni}) \to \sigma(k), \quad (1)$$

которая по существу является логическим определением класса (или соответствующего ему понятия). Из вышеизложенного ясно, что АК вида (1) могут строиться СИИ автоматически в режиме обучения по мере последовательного предъявления ей эталонов.

Практически важно, чтобы система АК обладала свойствами полноты, независимости и инвариантности в естественных смыслах. Дадим развернутое определение этих свойств.

Систему АК будем называть *полной* на множестве изображений {ω}, если для всякого изображения со из этого множества найдется АК, принимающая на нем значение «истинно». Следует отметить, однако, что полнота системы АК не исключает того, что для некоторого изображения могут найтись две АК, принимающие на нем значение «истинно».

В некоторых случаях требуется, чтобы ни одно исследуемое изображение не было отнесено одновременно к нескольким классам (например, если априори известно, что распознаваемые классы изображений не пересекаются). Это требование должно быть отражено в АК.

Систему АК будем называть *непротиворечивой*, если существует только одна АК, истинная на любом данном изображении. Из приведенного определения следует, что непротиворечивая система АК исключает возможность пересечения классов, описываемых этой системой аксиом. Очевидно, что непротиворечивость системы АК всегда можно эффективно проверить.

Вариативность изображений, порождаемая пространственными преобразованиями воспринимаемых предметов, а также действием разного рода помех и искажений, требует, чтобы система обладала определенной **инвариантностью и помехозащищенностью**. Например, всевозможные изображения стола, отличающиеся от эталонного (по которому строится соответствующая АК) сдвигом, поворотом, масштабом, а также некоторыми незначительными искажениями или помехами (лишние линии, незначительные изменения пропорций и т. п.), должны описываться одной и той же АК «стол», т. е. должны **классифицироваться как эквивалентные**.

Систему АК будем называть *инвариантной* по отношению к заданной группе преобразований, если каждая входящая в нее аксиома принимает одно и то же значение на изображениях, отличающихся преобразованиями из этой группы. Таким образом, инвариантность системы АК позволяет «снять» охарактеризованную выше

вариантность изображений и тем самым облегчить распознавание сцен по их изображениям.

Задачи распознавания и анализа изображений могут быть переформулированы как **задачи логического вывода**. **Эти задачи решаются в режиме распознавания**. Системе ИИ предъявляются сцена, изображение $\tilde{\omega}$ которой может содержать одно или несколько изображений предметов. Эти изображения отличаются от эталонов некоторыми преобразованиями и даже могут частично перекрываться. Описание изображения сцены $S(\tilde{\omega})$ представляет собой конъюнкцию всех первичных предикатов, истинных на данном изображении $\tilde{\omega}$. В этих условиях задача отыскания на изображении $\tilde{\omega}$ изображения из определенного класса Ω_k, т. е. задача распознавания, сводится к нахождению доказательства теоремы $S(\tilde{\omega}) \to \sigma(k)$. Саму эту теорему можно трактовать как вопрос: имеется ли на данном изображении $\tilde{\omega}$ предмет из *k*-го класса? Ясно, что в процессе ответа на этот вопрос описание $S(\tilde{\omega})$ может быть использовано не полностью. Поэтому **измерение тех или иных первичных признаков и предикатов должно производиться по мере необходимости в процессе логического вывода.**

Задача анализа изображения сцены заключается в распознавании на ней всех изображений предметов из различных классов. Формально эта задача сводится к последовательному доказательству теорем $S_i(\tilde{\omega}) \to \exists k \sigma(k)$, $i = 1, \ldots, N-1$, где $S_1(\tilde{\omega}) = S(\tilde{\omega})$, а $S_{i+1}(\tilde{\omega})$ получается из $S_i(\tilde{\omega})$ вычеркиванием всех предикатов, участвовавших в выводе *i*-й теоремы. Содержательно это означает, что, как только выделяется очередное изображение предмета из некоторого класса, оно при дальнейшем анализе не рассматривается. Полный анализ изображения сцены заканчивается распознаванием (и тем самым выделением) всех видимых изображений предметов, составляющих сцену.

В режиме обучения и распознавания на различных изображениях может встречаться один и тот же набор первичных признаков, характеризующих, например, фрагмент изображения. В таких случаях естественно ввести вторичные признаки, каждый из которых представляет собой некоторую совокупность из первичных признаков, а также вторичные предикаты, определяющие соответствующие фрагменты изображения как новые понятия.

Аксиомами обучения (АО) будем называть ППФ вида

$$\bigwedge_{j=1}^{r} \xi_j(x_{j1}, \ldots, x_{jm}) \to \alpha_i, \quad \textbf{(2)}$$

где $\xi_j(x_{j1}, \ldots, x_{jm}), j = 1, \ldots, r$ — первичные предикаты, дающие полное описание вторичного предиката α_i, т. е. определяющие некоторый фрагмент изображения как новое понятие. Использование АО позволяет не только более экономно представить АК, но и повысить эффективность системы логического вывода в процессе распознавания и анализа.

Универсальным средством логического вывода в исчислении предикатов является **метод резолюций**. Поэтому любой конкретный алгоритм распознавания или анализа определяется стратегией метода резолюций. Важно отметить, что для распознавания на изображении сцены нужного предмета не обязательно строить доказательство теоремы $S(\tilde{\omega}) \rightarrow \sigma(k)$ полностью, т. е. перебирать все элементы искомого простого изображения на изображении сцены $\tilde{\omega}$. Вместо этого достаточно найти фрагмент искомого изображения, который содержится лишь в изображениях *k*-го класса и не содержится ни в одном изображении предметов из других классов. Именно это обстоятельство позволяет сильно ограничить число шагов логического вывода, а также распознавать частично закрытые изображения предметов.

Качество работы алгоритмов распознавания и анализа естественно характеризовать числом обращений к информационно-измерительной (сенсорной) системе с целью определения нужных признаков. Заметим, что в нашей формализации это в точности совпадает с числом шагов логического вывода, т. е. **с числом резольвент, формируемых в процессе распознавания и анализа. Стратегию логического вывода будем называть *оптимальной,* если число шагов доказательства (число резольвент) минимально.**

Важно, отметить, что система логического вывода способна совершенствовать алгоритмы распознавания и анализа за счет использования элементов обучения, а именно: АО и адаптивной подстройки стратегии. Адаптивная подстройка стратегии осуществляется путем ее перестройки (например, путем перестройки оптимального распознающего графа) по мере распознавания новых изображений и формирования аксиом обучения (АО). Использование АО сокращает логический вывод на длину ее описания, а процесс построения оптимального распознающего графа — на значение экспоненциальной функции от этой длины.

Проиллюстрируем описанный метод на **примере решения задачи описания, распознавания и анализа обстановки в цехе.**

Предположим, что в результате предварительной фильтрации исходные изображения предметов и сцен превращаются в контурные

изображения, составленные из отрезков. Введем необходимые для дальнейшего первичные признаки и предикаты, представленные в табл. 1.

ТАБЛИЦА 1

Первичный признак	Предикат	Интерпретация
Вертикальный отрезок	$ВТ(x, y)$	Истинен, если точки x и y соединены вертикальным отрезком.
Горизонтальный отрезок	$ГР(x, y)$	Истинен, если точки x и y соединены горизонтальным отрезком.
Параллельность двух отрезков	$ПРЛ(v, y, u, v)$	Истинен, если отрезки (v, y) и (u, v) параллельны.

Вторичные признаки и предикаты (аксиомы обучения) представлены в табл. 2.

ТАБЛИЦА 2

Вторичный признак-понятие	Символ понятия	Логическое описание понятия
Параллелограмм	$ПР(x_1, x_2, x_3, x_4)$	$ПРЛ(x_1, x_2, x_3, x_4) \wedge ПРЛ(x_1, x_4, x_2, x_3)$
Горизонтальный параллелограмм	$ГПР(x_1, x_2, x_3, x_4)$	$ПР(x_1, x_2, x_3, x_4) \wedge ГР(x_1, x_2) \wedge ГР(x_2, x_3)$
Вертикальный прямоугольник	$ВПР(x_1, x_2, x_3, x_4)$	$ПР(x_1, x_2, x_3, x_4) \wedge ВТ(x_1, x_2) \wedge ГР(x_2, x_3)$

В режиме обучения системе ИИ предъявляются эталонные контурные изображения токарного и сверлильного станков, представленные на рис. 1.

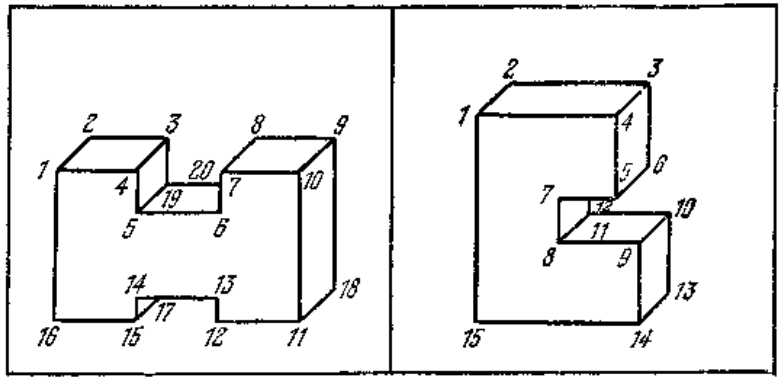


Рис. 1. Эталонные изображения (режим обучения).

По этим данным он строит АК, общий вид которых приведен в табл. 3.

ТАБЛИЦА 3

Класс-понятие	Символ класса	Логическое описание класса
Токарный станок	σ (*TC*)	$ГПР\ (x_1, x_2, x_3, x_4) \wedge ВПР\ (x_5, x_4, x_3, x_{19}) \wedge ГР\ (x_{19}, x_{20}) \wedge ГР\ (x_5, x_6) \wedge ВТ\ (x_6, x_7) \wedge ГПР\ (x_7, x_8, x_9, x_{10}) \wedge ВПР\ (x_{11}, x_{10}, x_9, x_{18}) \wedge ГР\ (x_{12}, x_{11}) \wedge ВТ\ (x_{12}, x_{13}) \wedge ГР\ (x_{14}, x_{13}) \wedge ВТ\ (x_{15}, x_{14}) \wedge ГР\ (x_{15}, x_{17}) \wedge ГР\ (x_{16}, x_{15}) \wedge ВТ\ (x_{16}, x_1)$
Сверлильный станок	σ (*CC*)	$ГПР\ (x_1, x_2, x_3, x_4) \wedge ВПР\ (x_5, x_4, x_3, x_6) \wedge ГР\ (x_7, x_5) \wedge ВТ\ (x_8, x_7) \wedge ГПР\ (x_6, x_{11}, x_{10}, x_9) \wedge ВТ\ (x_{11}, x_{12}) \wedge ВПР\ (x_{14}, x_9, x_{10}, x_{13}) \wedge ГР\ (x_{15}, x_{14}) \wedge ВТ\ (x_{15}, x_1)$

Так в режиме обучения автоматически формируется описание классов. В режиме распознавания сисиеме ИИ предъявляется изображение произвольной сцены — обстановки в цехе, попавшей в «поле зрения»СИИ. Для определенности пусть это будет изображение сцены, представленное на рис. 2.

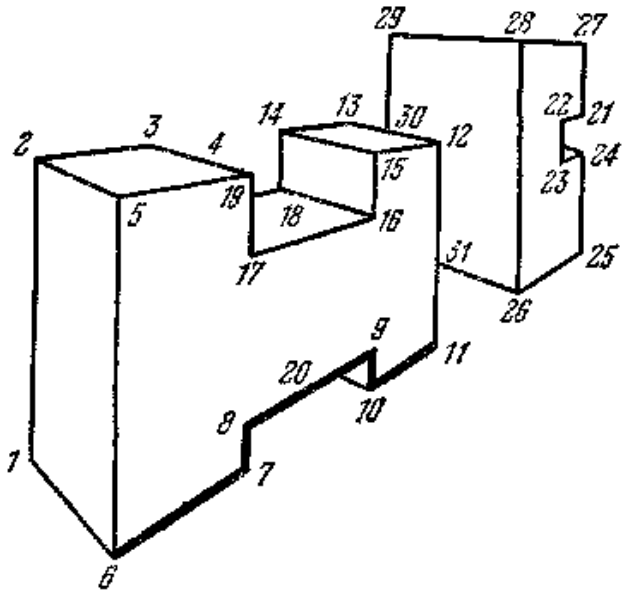


Рис.2. Изображение сцены (режим распознавания).

Описание этой сцены (на языке первичных и вторичных предикатов) имеет вид

1. ВПР (6, 5, 2, 1); 2. ГПР (5, 2, 3, 4); 3. ВТ (17, 4); 4. ГР (17, 16); 5. ВПР (16, 15, 14, 18); 6. ГР (19, 18); 7. ГПР (15, 14, 13, 12); 8. ВТ (11, 12); 9. ГР (10, 11); 10. ВТ (10, 9); 11. ГР (10, 20); 12. ГР (8, 9,); 13. ВТ (7, 8); 14. ГР (6, 7); 15. ВТ (30, 28); 16. ГР (29, 28); 17. ГР (29, 27); 18. ВТ (21, 27); 19. ГР (22, 21); 20. ВТ (23, 22); 21. ГР (23, 24); 22. ВТ (25, 24); 23. ГР (26, 25); 24. ВТ (26, 29); 25. ГР (26, 31).

Предположим, что мы спрашиваем СИИ (или она сама задается этим вопросом): «есть ли на изображении сцены токарный станок?». Для ответа на вопрос СИИ пытается доказать теорему

$$\bigwedge_{j=1}^{25} \xi_j\ (c_{j1}, \ \ldots\ , \ c_{jk}) \rightarrow \sigma\ (\mathrm{TC}),$$

где $\xi_j\ (c_{j1}, \ \ldots\ , \ c_{jk})$, $j = 1, \ \ldots\ , \ 25$,— предикаты, входящие в приведенное выше описание изображения сцены. Доказательство этой теоремы с помощью **стратегии лозы** потребовало в зависимости от порядка записи АК от 8 до 54 резольвент. Однако, как уже отмечалось, для распознавания токарного станка на изображении сцены полное доказательство соответствующей теоремы не обязательно, —

возможно распознавание по фрагменту. Использование оптимальной стратегии в этом примере позволило сократить число резольвент до 5. При этом был автоматически выделен и существенно использовался в процессе логического вывода фрагмент изображения, обведенный на рис. 2 жирной линией. Следует отметить, что решение той же задачи без использования АО приводит к существенному снижению эффективности системы логического вывода. В этом случае описание изображения сцены требует уже не 25, а 47 первичных предикатов. В процессе распознавания токарного станка с помощью стратегии лозы строится по меньшей мере 73 резольвенты.

3.2. Моделирование внешней среды

Любая система , вырабатывающая целесообразное поведение, — будь то человек, животное или СИИ, — **всегда использует знания о своих собственных функциональных возможностях и о свойствах внешней среды.** О таких системах будем говорить, что они обладают «**внутренней моделью внешнего мира**». Наличие модели среды позволяет быстро и глубоко проникать в механизмы внешних явлений, намного облегчает процессы обучения на опыте и адаптации.
Каковы же особенности и способы представления знаний о внешнем мире в памяти СИИ? Прежде всего заметим, что способность к моделированию внешней среды (включая свое собственное «я») присуща только СИИ. Обычные автоматические системы такой способностью не обладают.
Окружающий нас мир настолько сложен, что не только СИИ, но и человек бывает доволен, если ему удается уловить и понять хотя бы некоторые самые простые из присущих этому миру закономерностей. Для этого **человек строит упрощенные и идеализированные модели, освобожденные от маловажных подробностей и отражающие, как он надеется, наиболее существенные свойства рассматриваемых реальных объектов**. По такой же схеме должна действовать и СИИ , желающая создать в себе адекватную модель внешнего мира.
Восприятие внешнего мира осуществляется с помощью искусственных органов чувств СИИ. Это значит, что **реальные ситуации описываются в памяти СИИ с помощью набора показаний сенсорных датчиков.** Именно в **терминах этих показаний — элементарных высказываний о свойствах среды и самой СИИ — и формируется «внутренняя модель внешнего мира».**

Однако совокупность показаний сенсорных датчиков — это «сырая» информация, которую мы будем называть **первичным описанием реальной ситуации**. Анализ и обработка этой информации, как мы видели в предыдущем параграфе, позволяют строить обобщенные описания ситуаций, называемые понятиями. В эти информационные представления о мире могут входить не только показания органов чувств (например, «красное», «горячее», «твердое» и т. п.), но и утверждения о соотношениях между этими показаниями. Одним из наиболее удобных средств для формирования и обработки этих представлений **является исчисление предикатов вместе с адаптивной системой логического вывода**. На языке исчисления предикатов **свойства внешнего мира задаются с помощью элементарных предикатов и функций от показаний сенсорных датчиков, а также с помощью логических средств формирования из этих элементов сложных утверждений (описаний) в зависимости от имеющегося опыта и поступающей информации.** В предыдущем параграфе мы видели, каким образом осуществляется уточнение (коррекция) и расширение представлений СИИ о внешнем мире в процессе обучения на опыте и адаптации к существующим (заранее неизвестным) условиям.
Моделирование внешнего мира в памяти СИИ не является самоцелью. Оно служит главным образом для «мысленного» планирования поведения и принятия решений о тех или иных целенаправленных действиях СИИ . При этом весьма важно, чтобы в модели внешнего мира был отражена сама СИИ, ее структурные и функциональные свойства, особенности взаимодействия с окружающей средой. Благодаря этому СИИ получает возможность анализировать не только окружающую среду, в которой она функционирует, но и свое поведение в этой среде. Мы можем сказать, что СИИ обладает «сознанием» и «самосознанием». Уточним эти интуитивно понятные термины.
«Сознанием» СИИ будем называть ее способность отображать (моделировать) внешнюю среду в своей памяти и анализировать закономерности этой среды, а также результаты своих воздействий на среду. Под «самосознанием» будем понимать свойство СИИ отображать себя в модели среды и анализировать закономерности воздействия среды на свою структуру и функционирование.

3.3. Алгоритмы построения программных движений

Задача построения программных движений (ПД) СИИ решается в системе управления СИИ. Особенность этой задачи заключается в том, что движение исполнительных механизмов СИИ задается законом изменения обобщенных координат, а происходит в реальной внешней среде. При этом допустимость тех или иных конфигураций исполнительных механизмов непосредственно наблюдается в реальной же среде и никаким простым способом не выражается «на языке» пространства обобщенных координат.

Рассмотрим кратко некоторые идеи и алгоритмы построения ПД. Эти алгоритмы служат для того, чтобы СИИ могла, еще не совершая каких-либо реальных движений, «мысленно» рассчитать, а если нужно — скорректировать программу своих целенаправленных движений. В качестве примера рассмотрим задачу управления многозвенным манипулятором СИИ. Цель управления в этом случае заключается в отслеживании исполнительным механизмом СИИ некоторой траектории, рассчитанной на более высоком уровне планирования поведения. Задача осложняется наличием разного рода препятствий и конструктивных ограничений.

Программным движением называется такой закон изменения обобщенных координат, при котором достигается цель управления, удовлетворяются конструктивные ограничения и обеспечивается обход препятствий.

Идея «глобального» подхода к построению ПД заключается в следующем. Вначале на основе информации о траектории исполнительного механизма СИИ с учетом конструктивных ограничений и препятствий формируется *план движения* — последовательность промежуточных конфигураций, ведущая к цели. Затем по «кускам» (например, **в классе кусочно-полиномиальных функций**) строится само ПД в соответствии с этим планом.

Большой интерес представляет также «локальный» подход, основанный на моделировании так называемого тропизма. **Тропизм — это свойство, присущее простейшим живым организмам; оно заключается в направленных движениях организмов в результате действия односторонних раздражителей.** Применительно к уровню построения ПД **под таким «раздражителем» будем понимать информацию о взаимном расположении манипулятора и препятствий в реальном трехмерном пространстве, а под тропизмом — движения манипулятора, направленные на удаление**

от препятствий. Не приводя (ввиду громоздкости) явные формулы алгоритма построения ПД, имитирующего свойства тропизма, отметим только, что синтезируемые им целенаправленные движения реализуются за счет двигательной избыточности манипулятора, в то время как исполнительный механизм СИИ движется по заданной траектории.

Мы бегло охарактеризовали различные способы построения ПД манипулятора. А как строить ПД колесного или гусеничного шасси для СИИ, функционирующей на местности с препятствиями? Эта задача естественным образом распадается на две: прокладка безопасного маршрута и построение закона изменения обобщенных координат, удовлетворяющего конструктивным ограничениям и обеспечивающего движение по этому маршруту.

Рассмотрим один из подходов к построению *оптимального маршрута* на местности с препятствиями, основанный на методе динамического программирования. Для простоты предположим, что местность представляет собой плоскость, а препятствия задаются ломаными линиями. Пусть заданы координаты исходной точки (где находится СИИ) и целевой точки (куда она должна передвинуться). Задача заключается в построении маршрута (ломаной линии) из исходной точки в целевую, который не пересекает ломаных препятствий и имеет наименьшую длину. Такой маршрут будем называть оптимальным.

Прежде всего заметим, что если сформулированная задача не имеет тривиального решения (когда маршрутом является просто отрезок, соединяющий исходную и целевую точки), вершинами ломаной наименьшей длины должны быть вершины ломаных — препятствий. Поэтому в дальнейшем будем рассматривать только эти вершины.

Введем функцию f_i, определяющую минимальную длину пути из некоторой точки i в целевую точку. Сразу же возникает вопрос — как найти эту функцию? Один из классических способов нахождения функции состоит в том, чтобы задать уравнения, определяющие эту функцию.

Приступая к поиску такого уравнения, зададимся вопросом: существует ли такое внутреннее свойство процесса поиска оптимального пути, которое можно использовать для получения уравнения? Ответ, к счастью, положительный. В самом деле, анализируя задачу, мы видим, что отправляясь из точки i, СИИ на первом шаге попадет в точку j, пока неясно, какую именно. Далее, мы замечаем, что, если СИИ ищет кратчайший путь от точки i до целевой точки, то в какую бы точку она ни попала, путь из точки j в целевую должен иметь наименьшую длину. Это почти очевидное свойство докажем от противного. Пусть путь из точки i в целевую точку

является кратчайшим, тогда часть пути из точки j в целевую точку также должна иметь минимальную длину, так как, если бы эта часть пути не была кратчайшей, то ее можно было бы заменить более кратким путем и тем самым сократить общую длину пути, а это противоречит тому, что f_i по определению есть минимальная длина пути.

Таким образом, мы установили существенное внутреннее свойство процесса поиска оптимального маршрута: «хвост» (конец) процесса — это всегда кратчайший маршрут. Что все это дает для определения функции f_i ? Обозначим через s_{ij} — длину пути между точками i и j (хотя все еще неизвестно, что это за точка j), а через f_j — наименьшую длину пути между точкой j и целевой точкой. Выбирая в качестве точки j такую точку, которая минимизирует сумму $s_{ij} + f_j$, получаем уравнение

$$f_i = \min_{j \neq i} [s_{ij} + f_j]. \qquad (3)$$

Это — основное уравнение динамического программирования. Оно обладает тем свойством, что задает две функции f_i и j (i). В самом деле, если мы нашли f_i , то значение j, на котором достигается минимум (3), как раз и указывает ту точку, в которую СИИ нужно двигаться дальше. Важно отметить, что функция j (i) представляет собой по существу стратегию поиска оптимального маршрута, т. е. правило, которое позволяет СИИ, находящейся в произвольной точке i, определить, куда ей следует двигаться дальше.

Трудность решения уравнения (3) заключается в том, что неизвестная функция входит в обе части равенства. В такой ситуации приходится прибегать **к классическому методу последовательных приближений в функциональном пространстве**, используя рекуррентную формулу

$$f_i^{(k+1)} = \min_{j \neq i} [s_{ij} + f_j^{(k)}], \qquad (4)$$

где $f_i^{(k)}$ — k-е приближение искомой функции. Можно доказать сходимость алгоритма (4) для произвольном неотрицательной начальной функции f^0_i с единственным ограничением, что значение функции f в целевой точке равно нулю.

Решение уравнения (3) можно искать и в пространстве стратегий. Этот путь представляется более естественным: «мыслить» в терминах пространства стратегий СИИ более удобно, чем в терминах искусственно выбранных функций f_i. К тому же понятие стратегии сохраняет смысл и в тех случаях, когда функция не может быть определена. Какую же приближенную стратегию можно выбрать в задаче прокладки оптимального маршрута? Одна из возможных стратегий такова: СИИ решает непосредственно двигаться из точки i в

целевую точку. Тогда получается приближение $f^0_i = s_{iц}$, где $s_{iц}$ — длина пути между точкой i и целевой точкой. Следующее приближение получится, если СИИ будет искать решение в классе двухзвенных ломаных. Дальнейшие приближения ищутся в классе трехзвенных, четырехзвенных и т. д. ломаных.

Одно из преимуществ выбора приближений в пространстве стратегий состоит в следующем: выбрав из общих соображений стратегию, мы получаем соответствующую функцию $f^{(k)}_i$. Понятно, что $f^{(k)}_i \leq f_i$. В заключение, не входя в детали, отметим только, что описанный метод последовательных приближений в пространстве стратегий действительно сходится к решению уравнения (3) и допускает программную реализацию.

После того, как оптимальный безопасный маршрут построен, можно тем или иным методом (например, с **помощью кусочно-полиноминальной аппроксимации**) построить программное движение колесного или гусеничного шасси СИИ.

3.4. Алгоритмы адаптивного управления движением

После того как программное движение (ПД) СИИ построено, возникает следующая задача: найти закон управления исполнительными приводами, реализующий ПД. Решение этой задачи существенно осложняется *неопределенностью* условий функционирования СИИ. Природа этой неопределенности многообразна: изменение характеристик исполнительных двигателей и механизмов (например, в результате старения и износа), технологические допуски, возможные неисправности, а также изменение условий, внешних по отношению к СИИ, но оказывающих на нее полностью или частично неконтролируемые воздействия. Если степень неопределенности велика, то для построения законов управления ПД СИИ классические методы теории автоматического управления могут оказаться недостаточными. В подобных условиях даже широко используемые следящие приводы, параметры которых выбираются из априорных соображений о «средних» условиях функционирования, зачастую не обеспечивают реализацию ПД с требуемой точностью. Поэтому возникает необходимость в адаптивном управлении, восполняющем недостающую информацию в процессе функционирования СИИ.

Синтез *адаптивного управления ПД* осуществляется в два этапа. Вначале в предположении, что уравнения движения СИИ полностью известны, строится закон управления, обеспечивающий близость (с любой наперед заданной точностью) реального и программного движе-

ний. Эта «неадаптивная» задача легко решается методами классической теории управления. Однако воспользоваться таким «идеальным» законом управления практически нельзя, так как он зависит от ряда *варьируемых параметров* уравнения движения, значения которых неизвестны (варьируемыми параметрами могут быть, например, масса и моменты инерции объекта манипулирования, распределение нагрузки на шасси СИИ, коэффициенты сцепления с грунтом и т, п.). Поэтому на втором этапе на основе «идеального» закона строится адаптивное управление, обеспечивающее близость реального и ПД при любых возможных изменениях варьируемых параметров.

Идея адаптивного управления проста. Она заключается в замене неизвестных параметров «идеального» закона управления их оценками, которые должны целенаправленно «настраиваться» в процессе функционирования СИИ. Алгоритмы, по которым осуществляется «настройка» параметров управления, принято называть *алгоритмами адаптации.* Многие известные в настоящее время алгоритмы адаптации с математической точки зрения представляют собой разностные или дифференциальные уравнения, определяющие закон целенаправленного изменения параметров управления на основе сенсорной информации.

Примером эффективных алгоритмов адаптации, допускающих простую реализацию, могут служить *конечно-сходящиеся* алгоритмы решения целевых неравенств. Смысл целевых неравенств заключается в том, что если они выполнены, то закон управления, в котором вместо неизвестных параметров используется их оценка, обеспечивает требуемую близость реального и ПД. В этом случае коррекция («настройка») параметров не производится. Если же целевые неравенства в некоторый момент времени нарушаются, то осуществляется адаптивная коррекция параметров по некоторому алгоритму. Этот алгоритм может быть выбран так, что число коррекций будет конечным (и даже минимально возможным).

Описанный конечно-сходящийся процесс адаптации хорошо согласуется с содержательным представлением об адаптации. В самом деле, с интуитивной точки зрения адаптация должна проявляться в том, что в неизменяющихся («стационарных») условиях функционирования СИИ алгоритм адаптации «работает» не все время, а с течением времени «отключается», осуществляя переход на автоматическое управление без «настройки» параметров. Лишь значительное изменение условий функционирования СИИ вызывает необходимость «включения» алгоритма адаптации для коррекции параметров управления.

4. Предмет и задачи компьютерной обработки и распознавания изображений

4.1 Определение компьютерной обработки изображений

Компьютерная обработка и распознавание изображений представляет собой самостоятельную дисциплину.

Компьютерная обработка изображений предполагает обработку цифровых изображений с помощью компьютеров или специализированных устройств, построенных на цифровых сигнальных процессорах.

При этом **под обработкой изображений** понимается не только улучшение зрительного восприятия изображений, но и классификация объектов, выполняемая при анализе изображений.

В 60-е годы прошлого века получила развитие особая наука об изображениях - **«иконика», которая посвящена исследованиям общих свойств изображений, целей и задач их преобразования, обработки и воспроизведения, распознавания графических образов**. Термин «иконика» происходит от греческого «eikon», что означает изображение, образ. Сегодня под ним понимают «создание и обработку изображений с помощью ЭВМ, что совпадает с понятием компьютерной обработки изображений.

Области применения цифровой обработки значительно расширяются, вытесняя аналоговые методы обработки сигналов изображений. Методы цифровой обработки широко применяются в промышленности, искусстве, медицине, космосе и, особенно, в искусственном интеллекте. Они применяются при управлении процессами, автоматизации обнаружения и сопровождения объектов, распознавании образов и во многих других приложениях. Цифровая передача изображений с космических аппаратов, цифровые каналы передачи сигналов изображений требуют обеспечения передачи все больших потоков информации. Если при передаче цифрового сигнала цветного телевидения необходимо передавать потоки порядка 216 Мбит/с, то для передачи телевидения высокой четкости скорость передачи должна составлять порядка 1 Гбит/с. Формирование изображений, улучшение качества и автоматизация обработки изображений, включая изображения, создаваемые электронными

микроскопами, рентгеновскими аппаратами, томографами и т.д., являются предметом широкого исследования и разработки. В системах искусственного интеллекта широко применяются системы формирования изображения, его преобразования в цифровую форму, визуализация и документирование путем введения в компьютер изображений с помощью специализированных устройств захвата видео. Автоматический анализ в системах дистанционного наблюдения широко применяется при анализе местности, в лесном хозяйстве, например, для автоматического подсчета площади вырубок, в сельском хозяйстве для наблюдения за созреванием урожая, при разведке, в системах противопожарной безопасности. Контроль качества производимой продукции выполняется благодаря автоматическим методам анализа сцен. Компьютерная обработка изображений применяется в задачах экспертизы живописи неразрушающими методами. Для восстановления старых фильмов применяются методы автоматической компенсации дефектов видеоматериала, полученного после преобразования киноизображения в видео.

Трудно представить область деятельности, в которой можно обойтись без компьютерной обработки изображений. Интернет, сотовый телефон, видеокамера, фотоаппарат, сканер, принтер, так прочно вошедшие в наш быт, - немыслимы без компьютерной обработки изображений.

При компьютерной обработке изображений решается широкий круг задач, таких как улучшение качества изображений; измерение параметров; спектральный анализ многомерных сигналов; распознавание изображений; сжатие изображений.

Устройства формирования изображений получили широкое распространение и применение в самых различных областях науки, техники, промышленности, медицине, биологии и др. Они являются неотъемлемыми компонентами систем и устройств, применяемых в фотокинотехнике, телевидении, системах технического зрения: дневного, ночного и теплового видения, при дистанционном зондировании Земли. Назначение этих систем предполагает решение комплекса технических и научных задач, требующих синтеза и анализа методов обработки, бинаризации, классификации изображений. Развитие микроэлектроники, переход от аналоговой формы сигналов к цифровой позволяют расширить палитру и повысить сложность применяемых алгоритмов для решения поставленных задач. Рассмотрим некоторые из устройств формирования изображений.

4.2. Основные задачи обработки изображений

Решение многих проблем искусственного интеллекта приводит к необходимости извлечения полезной информации из различного рода многомерных данных, которые, по аналогии с оптическими изображениями, будем называть **многомерными изображениями** (МИ) или просто **изображениями** (И). Такие задачи возникают в очень многих областях знаний: в медицине, радио-, тепло- и гидролокации, исследовании Космоса и Земли, телевидении и т. д. Например, диагностика различных заболеваний по И внутренних органов человека, обнаружение лесных пожаров, поиск перспективных для ловли рыбы акваторий, оценка экологического состояния регионов, навигационные задачи и т. д.

Характерно, что эти задачи приходится решать при наличии различного рода мешающих факторов – помех, мешающих И, переменчивости условий наблюдения, динамики наблюдаемого объекта, взаимного перемещения приемника и объекта и т. п. Полезный сигнал может быть очень слаб по отношению к помехам и визуально неразличим на фоне мешающих И.

Нередко объем исходных данных очень велик (глобальный мониторинг Земли, массовые медицинские обследования), они поступают с большой скоростью и требуют обработки в режиме реального времени. Оператор не в состоянии справиться с таким потоком информации.

Единственным выходом из такой ситуации является компьютерная обработка И. Для этого необходимо создание соответствующих математических методов описания и обработки И, а также программного обеспечения применительно к конкретным задачам.

Несмотря на огромное разнообразие практических задач обработки И, они сводятся к небольшому количеству следующих основных задач.

1) **Фильтрация и улучшение визуального восприятия.** Как уже отмечалось, полезное И может наблюдаться на фоне различных помех, которые и требуется по возможности ослабить. Кроме того, может потребоваться сделать И более контрастным, выделить контуры и т. д.

2) **Восстановление отсутствующих участков.** Из-за сбоев передачи И или особо сильных помех отдельные участки И могут отсутствовать. Задача заключается в их восстановлении. Такая задача возникает, например, при реставрации картин, фотографий и фильмов.

3) **Обнаружение объектов и их идентификация.** Требуется на фоне мешающих И найти интересующие нас объекты. Если таких объектов может быть несколько типов, то дополнительно нужно их классифицировать. В качестве примеров можно привести автоматическое считывание номеров проезжающих автомобилей, обнаружение и идентификацию летательных аппаратов, обнаружение лесных пожаров и т. д. Иногда задача обнаружения ставится менее определенно (найти то, не знаю что) – требуется обнаружить аномалии, т. е. участки И, чем-то отличающиеся от своего окружения. Например, к таким отличиям может привести наличие полезных ископаемых, сельскохозяйственных вредителей или локальных патологий внутренних органов.

4) **Оценка геометрических трансформаций и совмещение И.** В процессе наблюдения все И или отдельные его части могут перемещаться из-за динамики сцены, движения приемника или несовершенства его конструкции, турбулентности атмосферы и т. д. В результате одни и те же элементы И находятся на наблюдаемых кадрах в разных местах, т.е. имеются геометрические трансформации И. Иногда эти трансформации являются мешающим фактором, например, динамика медицинских И при дыхании пациента. В других случаях трансформации – фактор информативный, например, по изображениям движущихся облаков можно оценить поле скоростей ветра в окрестности аэропорта, что нужно для обеспечения безопасности полетов. В любом случае требуется оценить геометрические трансформации, т. е. совместить элементы одного И с соответствующими им элементами на другом И.

5) **Оценка параметров И.** В эту группу входят задачи измерения различных характеристик И или их отдельных элементов: вероятностные характеристики И, положение и размеры объектов и т. д.

6) **Сжатие И.** Большой объем и высокая скорость поступления данных ставят повышенные требования к накопителям и каналам передачи И. Использование специфики И часто дает возможность

достигнуть значительно большего сжатия, чем это позволяют обычные архиваторы.

Кроме перечисленных задач собственно обработки И, иногда возникает задача их понимания, т. е. «а что бы это значило и какие отсюда следуют выводы?»

Исследования по математическому описанию и анализу И были начаты в 50-х годах 20-го века, но только в 70-х годах эти исследования привлекли большое внимание ученых во всем мире и стали проводиться очень интенсивно и по широкому фронту, что вызвано необходимостью автоматического анализа И в различных приложениях. Кроме того, появление прогресс вычислительной техники позволяет реализовать в реальном времени более эффективные алгоритмы обработки И, чем это было возможно ранее.

5. Математические модели изображений

5.1. Модели непрерывных изображений

Компьютерная обработка изображений возможна после преобразования сигнала изображения из непрерывной формы в цифровую форму. Эффективность обработки зависит от адекватности модели, описывающей изображение, необходимой для разработки алгоритмов обработки. При этом необходимо учитывать влияние передающей и приемной систем и канала связи на сигнал изображения. **Модель изображения представляет систему функций, описывающих существенные характеристики изображения: функцию яркости, отражающую изменение яркости в плоскости изображения, пространственные спектры и спектральные интенсивности изображений, функции автокорреляции. Канал изображения содержит оптическую систему, оптико - электрический преобразователь, устройство аналого - цифрового преобразования (АЦП) и цифровой обработки сигналов изображения.** В общем случае непрерывное изображение может быть представлено функцией **пяти аргументов**: **трех пространственных координат, времени и длины волны электромагнитного излучения**. Упрощения модели пространственно - временных сигналов в некотором диапазоне волн $C(x,y,z,t,\lambda)$ приводят к

моделям пространственно- временного сигнала *C(x,y,z,t),* пространственного сигнала *C(x,y,z),* временного сигнала *C*(*t*) . Здесь *x,y,z* -пространственные координаты, *t* - время, λ - длина волны электромагнитного излучения.

5.2. Представление непрерывных изображений

Распределение энергии источника светового излучения по пространственным координатам x,y, времени t и длинам волн λ описывается функцией $C(x,y,t,\lambda)$. Энергия излучения пропорциональна квадрату амплитуды электрического поля и поэтому представляет собой действительную положительную величину. В изображающих системах максимальная яркость изображения ограничена из-за насыщения светочувствительной пленки или перегревания люминофора кинескопа. Следовательно,

$$0 \le C(x,y,t,\lambda) \le A \qquad (2.1)$$

где A — максимальная яркость изображения. Размеры изображения ограничены формирующей системой и средой, на которую оно записываемся. В целях упрощения будем считать, что все изображения отличны от нуля только в прямоугольной области, для которой

$$-L_x \le x \le L_x \qquad (2.2а)$$

$$-L_y \le y \le L_y \qquad (2.2б)$$

Изображение наблюдается в течение конечного промежутка времени, т. е.

$$-T \le t \le T \qquad (2.2в)$$

Таким образом, величина $C(x,y,t,\lambda)$ является **ограниченной функцией** четырех ограниченных переменных. Будем считать эту функцию непрерывной в области ее определения.

Ощущение светлоты, возникающее в зрительной системе человека, обычно определяется мгновенной яркостью светового поля, т. е. величиной

$$Y(x,y,t) = \int_0^\infty C(x,y,t,\lambda) V_\lambda(\lambda) d\lambda \qquad (2.3),$$

где $V_\lambda(\lambda)$ — спектральная чувствительность человеческого зрения. Цветовые ощущения можно описать набором так называемых координат цвета, пропорциональных интенсивностям красного, зеленого и синего цвета, смесь которых дает заданный цвет. Для произвольной красно-зелено-синей координатной системы текущие значения координат цвета равны

$$R(x,y,t) = \int_0^\infty C(x,y,t,\lambda) R_\lambda(\lambda) d\lambda \qquad (2.4а),$$

$$G(x,y,t) = \int_0^\infty C(x,y,t,\lambda) G_\lambda(\lambda) d\lambda \qquad (2.4б),$$

$$B(x,y,t) = \int_0^\infty C(x,y,t,\lambda) B_\lambda(\lambda) d\lambda \qquad (2.4в),$$

где $R_\lambda(\lambda)$, $G_\lambda(\lambda)$, $B_\lambda(\lambda)$ — удельные координаты для набора основных цветов — красного, зеленого и синего, равные координатам цвета монохроматического излучения единичной интенсивности с длиной волны λ. В спектрозональных системах изображение для i-й спектральной зоны описывается выражением

$$F_i(x,y,t)=\int_0^{\infty} C(x,y,t,\lambda)S_i(\lambda)\,d\lambda, \qquad (2.5),$$

где $S_i(\lambda)$ — спектральная чувствительность i-го датчика.

Для простоты будем во всех случаях описывать изображение, сформированное некоторой физической системой, с помощью функции $F(x,y,t)$. Для одноцветной системы функция $F(x,y,t)$ представляет распределение яркости или какой-либо другой физической величины, связанной с яркостью. Для системы воспроизведения цветных изображений $F(x,y,t)$ есть одна из координат цвета. Функция $F(x,y,t)$ используется также для описания других полей, например меняющегося во времени пространственного распределения шума видеодатчика.

В соответствии с обычным определении среднего значения одномерного сигнала средняя по времени яркость изображения в данной точке x,y определяется как

$$\langle F(x,y,t)\rangle_T=\lim_{T\to\infty}\left\{(1/2T)\int_{-T}^{T}F(x,y,t)L(t)dt\right\}, \qquad (2.6)$$

где $L(t)$ — временная весовая функция. Аналогично определяется пространственная средняя яркость в момент времени t:

$$\langle F(x,y,t)\rangle_s=\lim_{\substack{L_x\to\infty\\ L_y\to\infty}}\left\{(1/4L_xL_y)\int_{-L_x}^{L_x}\int_{-L_y}^{L_y}F(x,y,t)dxdy\right\}. \qquad (2.7)$$

Во многих системах воспроизведения изображений, например в проекционных устройствах, изображение не изменяется во времени и

переменная t может быть опущена. В системах другого типа, например кинематографических, аргумент t меняется дискретно. В дальнейшем аргумент t функции, списывающей изображения, будет обычно опускаться.

5.3. Двумерные системы

В двумерной системе набор исходных двумерных функций $F_1(x,y),F_2(x,y),\ \dots\ ,F_N(x,y)$ отображается в набор двумерных функций $G_1(x,y),G_2(x,y),\ \dots\ ,G_M(x,y)$, где x,y — независимые непрерывные пространственные переменные, аргументы этих функций $(-\infty<x,y<\infty)$. Это отображение может быть задано набором операторов $O_m(\cdot)$ при $m=1,2,\dots,M$, которые связывают входные и выходные функции:

$$\begin{aligned}&G_1(x,y)=O_1\{F_1(x,y),F_2(x,y),\ \dots\ ,F_N(x,y)\},\\ &\dots\dots\dots\dots\dots\dots\dots\dots\dots\dots\dots\dots\dots\dots\dots\dots\dots\dots\dots\dots\\ &G_M(x,y)=O_M\{F_1(x,y),F_2(x,y),\ \dots\ ,F_N(x,y)\}.\end{aligned} \qquad (3.1)$$

Число входных функций N может быть больше, меньше или равняться числу выходных. При $N=M=1$

$$G(x,y)=O(F(x,y)). \qquad (3.2)$$

В одномерных физических системах, в которых независимой переменной является время, выходной сигнал представляет собой функцию прошлых и нестоящих значений входного сигнала, но не может быть функцией будущих значений. Такие системы, называемые физически реализуемыми, удовлетворяют принципу причинности. Двумерные системы в общем случае не обладают этим свойством: пространственные переменные x,y могут принимать отрицательные значения по отношению к некоторому началу координат.

Чтобы продолжить обсуждение свойств двумерных систем, необходимо познакомиться с некоторыми специальными видами операторов.

5.4. Сингулярные операторы

Сингулярные операторы широко применяются при анализе двумерных систем, особенно систем, в которых производится дискретизация непрерывных функций. Двумерная дельта-функция Дирака есть сингулярный оператор, обладающий следующими свойствами:

$$\delta(x,y)=\begin{cases}\infty, & x=0,y=0,\\ 0, & \text{в остальных случаях,}\end{cases} \tag{4.1а}$$

$$\delta(x-\xi,y-\eta)=\begin{cases}\infty, & x=\xi,y=\eta,\\ 0, & \text{в остальных случаях,}\end{cases} \tag{4.1б}$$

$$\int_{-\varepsilon}^{\varepsilon}\int \delta(x,y)\,dx\,dy=1 \quad \text{при} \quad \varepsilon>0, \tag{4.1в}$$

$$\int_{-\infty}^{\infty}\int F(\xi,\eta)\delta(x-\xi,y-\eta)\,d\xi\,d\eta=F(x,y). \tag{4.1г}$$

Величина ε здесь обозначает бесконечно малый предел интегрирования.

Двумерная дельта-функция может быть представлена как произведение двух одномерных дельта-функций ортогональных координат x,y:

$$\delta(x,y)=\delta(x)\,\delta(y), \tag{4.2}$$

где одномерные дельта-функции удовлетворяют одномерным соотношениям, аналогичным (4.1). Дельта-функция может быть также определена как предел некоторых функций например прямоугольной функции

$$\delta(x,y)=\lim_{a\to\infty}[a^2\,\mathrm{rect}(ax)\,\mathrm{rect}(ay)], \tag{4.3а}$$

круговой функции

$$\delta(x,y)=\lim_{a\to\infty}[(a^2/\pi)\,\mathrm{cirk}(a\sqrt{x^2+y^2})], \tag{4.3б}$$

гауссовой функции

$$\delta(x,y)=\lim_{a\to\infty}[a^2\exp(-a^2\pi(x^2+y^2))], \tag{4.3в}$$

sinc-функции

$$\delta(x,y)=\lim_{a\to\infty}[a^2\,\mathrm{sinc}(ax)\,\mathrm{sinc}(ay)], \tag{4.3г}$$

бесселевой функции

$$\delta(x,y)=\lim_{a\to\infty}\left[\frac{aJ_2(2\pi a\sqrt{x^2+y^2})}{\sqrt{x^2+y^2}}\right], \tag{4.3д}$$

где

$$\mathrm{rect}(x)=\begin{cases}1, & |x|\le 1/2,\\ 0, & |x|>1/2.\end{cases} \tag{4.4а}$$

$$\mathrm{circ}(r) = \begin{cases} 1, & r \le 1, \\ 0, & r > 1, \end{cases} \tag{4.4б}$$

$$\mathrm{sinc}(x) = \sin \pi x / \pi x. \tag{4.4в}$$

Другим полезным определением дельта-функции является следующее тождество:

$$\delta(x-\xi, y-\eta) = (1/4\pi^2) \int\int_{-\infty}^{\infty} \exp\{i[u(x-\xi)+v(y-\eta)]\}\, du\, dv, \tag{4.5}$$

где $i = \sqrt{-1}$.

5.5. Лнейные операторы

Двумерная система называется линейной, если для нее справедлив принцип суперпозиции. В частном случае отображения функции в функцию для этого требуется, чтобы

$$O\{a_1 F_1(x,y) + a_2 F_2(x,y)\} = a_1 O\{F_1(x,y)\} + a_2 O\{F_2(x,y)\}, \tag{5.1}$$

где a_1, a_2 — некоторые постоянные (могут быть комплексными). Определение свойства суперпозиции можно легко распространить на отображение (3.1) общего вида.

Используя свойство дельта-функции (4.1г), функцию на входе системы $F(x,y)$ можно представить как взвешенную сумму дельта-функций:

$$F(x,y) = \int\int_{-\infty}^{\infty} F(\xi,\eta)\, \delta(x-\xi, y-\eta)\, d\xi\, d\eta, \tag{5.2}$$

где $F(\xi,\eta)$ — весовой множитель дельта импульса, имеющего координаты ξ,η на плоскости (x,y) (рис.5.1). Если функция на выходе линейной системы

$$G(x,y) = O\{F(x,y)\}, \tag{5.3}$$

то

$$G(x,y) = O\left\{\int\int_{-\infty}^{\infty} F(\xi,\eta)\delta(x-\xi, y-\eta)\, d\xi\, d\eta\right\}, \tag{5.4а}$$

или

$$G(x,y) = \int\int_{-\infty}^{\infty} F(\xi,\eta)\, O\{\delta(x-\xi, y-\eta)\}\, d\xi\, d\eta. \tag{5.4б}$$

Для перехода от выражения (5.4а) к (5.4б) был изменен порядок выполнения операций линейного преобразования и интегрирования. Линейный оператор действовал только на тот множитель подынтегрального выражении (2.4а), который зависит от пространственных переменных x,y. Запишем второй множитель подынтегрального выражения (5.4б) как

$$H(x,y;\xi,\eta) = O\{\delta(x-\xi, y-\eta)\}. \tag{5.5}$$

Будем называть эту функцию импульсным откликом двумерной системы. Импульсный отклик оптической системы часто называется функцией рассеяния точки.

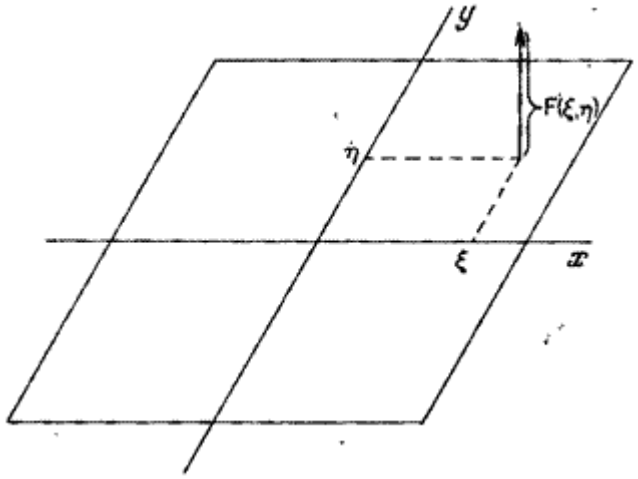


Рис.5.1. Представление функции, описывающей изображение, в виде суперпозиции дельта-функций.

Подстановка импульсного отклика в соотношение (5.4б) дает интеграл суперпозиции

$$G(x,y) = \int_{-\infty}^{\infty}\int F(\xi,\eta)\, H(x,y,\xi,\eta)\, d\xi\, d\eta. \qquad (5.6)$$

Линейная двумерная система называется пространственно-инвариантной (изопланатической), если ее импульсный отклик зависит только от разностей координат $x-\xi, y-\eta$. Для оптической системы, показанной на рис.5.2. это значит, что при перемещении точечного источника в предметной плоскости изображение этого источника в плоскости фокусировки будет также изменять положение, но сохранять форму. Для пространственно-инвариантной системы

$$H(x,y,\xi,\eta) = H(x-\xi, y-\eta) \qquad (5.7)$$

и интеграл суперпозиции имеет особую форму, называемую интегралом свертки:

$$G(x,y) = \int_{-\infty}^{\infty}\int F(\xi,\eta)\, H(x-\xi, y-\eta)\, d\xi\, d\eta. \qquad (5.8а)$$

Операции свертки символически записывается как

$$G(x,y) = F(x,y) * H(x,y). \qquad (5.8б)$$

Интеграл свертки симметричен, т. е.

$$G(x,y) = \int_{-\infty}^{\infty}\int F(x-\xi, y-\eta)\, H(\xi,\eta)\, d\xi\, d\eta. \qquad (5.9)$$

Процесс свертки иллюстрируется на рис.5.3. На рис.5.3, а и 5.3, б изображены функция $F(x,y)$ на входе и импульсный отклик.

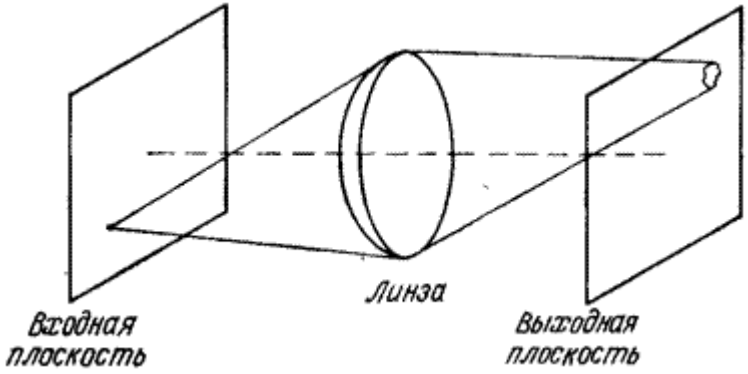


Рис.5.2. Изображение точечного источника света в оптической системе.

На рис.5.3, в показан импульсный: отклик при обращении координат, а на рис.5.3, г — со сдвигом на величину x, y. На рис.4.3, д заштрихована область, в которой произведение

$F(\xi,\eta)\,H(x-\xi,y-\eta)$, входящее в подынтегральное выражение (5.8, а), не равно нулю. Интегрирование на 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.

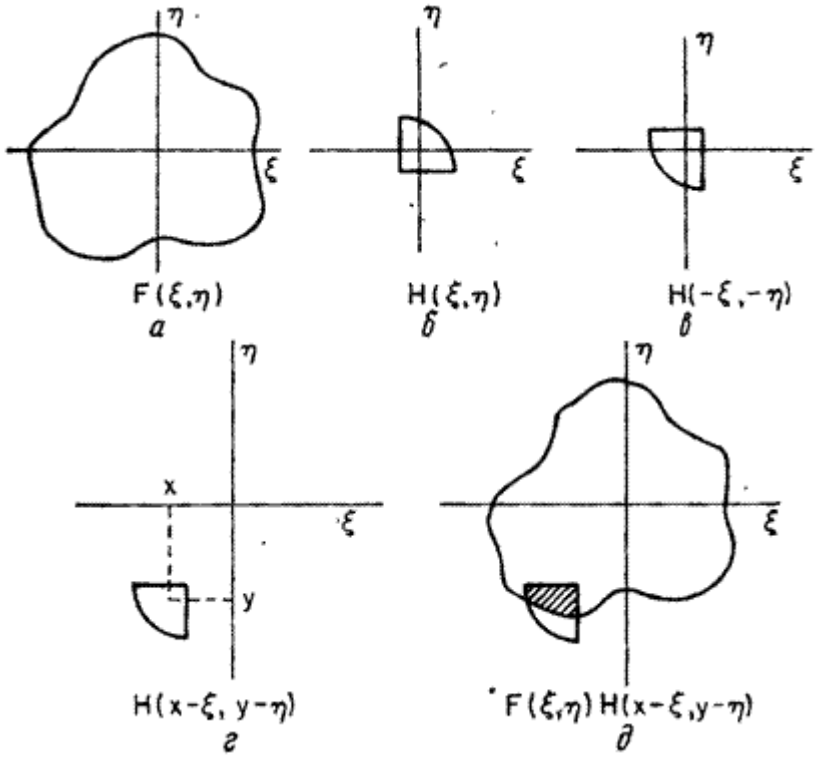


Рис.5.3. Пример двумерной свертки.

5.6. Дифференциальные операторы

Обнаружение и выделение контуров (границ) на изображении т.е. точек с резкими перепадами яркости, обычно осуществляются дифференцированием функции $F(x,y)$ по пространственным координатам и сравнением результата дифференцирования с порогом.

Первые производные по пространственным координатам x и y определяются как

$$d_x = \partial F(x,y)/\partial x, \quad (6.1а)$$

$$d_y = \partial F(x,y)/\partial y. \quad (6.1б)$$

Производная по направлению z, составляющему угол Φ с горизонтальной осью, по определению [3, стр. 10] равна

$$\nabla(F(x,y)) = \partial F(x,y)/\partial z = d_x \cos\Phi + d_y \sin\Phi. \quad (6.2)$$

Модуль градиента есть

$$|\nabla(F(x,y)| = \sqrt{d_x^2 + d_y^2}. \quad (6.3)$$

Вторые производные по пространственным координатам x и y по определению равны

$$d_{xx} = \partial^2 F(x,y)/\partial x^2, \quad (6.4а)$$

$$d_{yy} = \partial^2 F(x,y)/\partial y^2. \quad (6.4б)$$

Сумма производных (6.4) дает лапласиан

$$L(F(x,y)) = d_{xx} - d_{yy} = \partial^2 F(x,y)/\partial x^2 + \partial^2 F(x,y)/\partial y^2 \quad (6.5)$$

Следует отметить, что лапласиан является скалярной величиной, которая не зависит от направления на плоскости x, y, тогда как градиент есть зависящая от направления векторная величина.

5.7. Двумерное преобразование Фурье

В результате двумерного преобразования Фурье функции $F(x,y)$, описывающей изображение, получается спектр этого изображения, который определяется как

$$\Phi(\omega_x, \omega_y) = \int_{-\infty}^{\infty}\int F(x,y)\exp\left[-i(\omega_x x + \omega_y y)\right]dxdy \quad (7.1)$$

где ω_x, ω_y - пространственные частоты, а $i = \sqrt{-1}$. Если обозначить оператор преобразования Фурье через $O_F(\cdot)$, то можно записать

$$\Phi(\omega_x, \omega_y) = O_F(\Phi(x,y)) \quad (7.2)$$

В общем случае спектр $\Phi(\omega_x, \omega_y)$ есть комплексная величина. Его можно разложить на действительную и мнимую части:

$$\Phi(\omega_x, \omega_y) = R(\omega_x, \omega_y) + iI(\omega_x, \omega_y) \quad (7.3а)$$

или представить с помощью амплитуды и фазы:

$$\Phi(\omega_x, \omega_y) = M(\omega_x, \omega_y)\exp(i\phi(\omega_x, \omega_y)), \quad (7.3б)$$

где

$$M(\omega_x, \omega_y) = [R^2(\omega_x, \omega_y) + I^2(\omega_x, \omega_y)]^{1/2}, \quad (7.4а)$$

$$\phi(\omega_x, \omega_y) = \operatorname{arctg}(I(\omega_x, \omega_y)/R(\omega_x, \omega_y)). \quad (7.4б)$$

Достаточным условием существования фурье-спектра функции $\Phi(x,y)$ является абсолютная интегрируемость этой функции, т.е. условие

$$\int_{-\infty}^{\infty}\int |F(x,y)|dxdy < \infty. \quad (7.5)$$

Исходная функция $F(x,y)$ может быть восстановлена обратным преобразованием Фурье:

$$F(x,y) = (1/4\pi^2)\int_{-\infty}^{\infty}\int \Phi(\omega_x, \omega_y)\exp[i(\omega_x x + \omega_y y)]d\omega_x d\omega_y. \quad (7.6а)$$

Это соотношение в операторной форме можно записать как

$$F(x,y) = O_F^{-1}(\Phi(\omega_x, \omega_y)). \quad (7.6б)$$

Поскольку ядро двумерного преобразования Фурье разделимо, это преобразование может быть выполнено в два этапа. Сначала находится

$$\Phi_x(\omega_x, y) = \int_{-\infty}^{\infty} F(x,y)\exp(-i\omega_x x)dx, \quad (7.7)$$

а затем

$$\Phi(\omega_x, \omega_y) = \int_{-\infty}^{\infty} \Phi_y(\omega_x, y) \exp(-i\omega_y y)\,dy. \tag{7.8}$$

Ниже приводятся несколько свойств двумерного преобразования Фурье.

Функциональные свойства

Если функция $F(x,y)$ разделима по пространственным переменным, так что

$$F(x,y) = f_x(x) f_y(y), \tag{7.9}$$

то

$$\Phi(\omega_x, \omega_y) = \Psi_x(\omega_x)\Psi_y(\omega_y), \tag{7.10}$$

где $\Psi_x(\omega_x)\Psi_y(\omega_y)$ - одномерные фурье-спектры функций $f_x(x)$, $f_y(y)$. Если $\Phi(\omega_x, \omega_y)$ есть фурье-спектр функции $F(x,y)$, то $\Phi^*(-\omega_x, -\omega_y)$ является фурье-спектром функции $F^*(x,y)$. (Звёздочка обозначает комплексную сопряженность.) Если функция $F(x,y)$ симметрична, т.е. $F(x,y) = F(-x,-y)$, то $\Phi(\omega_x, \omega_y) = \Phi(-\omega_x, -\omega_y)$.

Линейность

Оператор преобразования Фурье линеен:

$$O_\Phi[aF_1(x,y) + bF_2(x,y)] = a\Phi_1(\omega_x, \omega_y) + b\Phi_2(\omega_x, \omega_y), \tag{7.11}$$

где a, b – постоянные.

Изменение масштаба

Изменение масштаба пространственных приводит к обратному изменению масштаба пространственных частот и пропорциональному изменению значений спектра:

$$O_\Phi(F(ax,by)) = \left(1/|ab|\right)\Phi\left(\omega_x/a, \omega_y/b\right). \tag{7.12}$$

Следовательно, сжатие вдоль одной из осей плоскости (x,y) приводит к растяжению вдоль соответствующей оси частотной плоскости и наоборот. Происходит также пропорциональное изменение значений спектра.

Сдвиг

Сдвиг (изменение координат) на исходной плоскости приводит к фазовым изменения на частотной плоскости:

$$O_\Phi(F(x-a, y-b)) = \Phi(\omega_x, \omega_y)\exp(-i(\omega_x a + \omega_y b)) \tag{7.13а}$$

Наоборот, сдвиг на частотной плоскости вызывает фазовые изменения исходной функции:

$$O_\Phi^{-1}(\Phi(\omega_x - a_x, \omega_y - b_y)) = F(x,y)\exp(i(a_x x + b_y y)). \tag{7.13б}$$

Свертка

Фурье-спектр функции, полученный в результате свертки двух функций, равен произведению спектров исходных функций:

$$O_\Phi\{F(x,y)*H(x,y)\} = \Phi(\omega_x,\omega_y)\mathcal{H}(\omega_x,\omega_y).$$

(7.14)

Обратная теорема утверждает, что

$$O_\Phi\{F(x,y)H(x,y)\} = (1/4\pi^2)\Phi(\omega_x,\omega_y)*\mathcal{H}(\omega_x,\omega_y).$$

(7.15)

Теорема Парсеваля

Два представления энергии изображения – через функцию $F(x,y)$ и фурье-спектр $\Phi(\omega_x,\omega_y)$ - связаны следующим образом:

$$\int_{-\infty}^{\infty}\int|F(x,y)|^2\,dxdy = (1/4\pi^2)\int_{-\infty}^{\infty}\int|\Phi(\omega_x,\omega_y)|^2\,d\omega_x d\omega_y.$$

(7.16)

Теорема о спектре автокорреляционной функции

Фурье-спектр двумерной автокорреляционной функции изображения равен квадрату модуля фурье-спектра этого изображения:

$$O_\Phi\left\{\int_{-\infty}^{\infty}\int F(\alpha,\beta)F^*(\alpha-x,\beta-y)\,d\alpha d\beta\right\} = |\Phi(\omega_x,\omega_y)|^2.$$

(7.17)

Спектры пространственных производных

Фурье-спектры первых пространственных производных функции $F(x,y)$ связаны с её фурье-спектром следующими соотношениями:

$$O_\Phi\{\partial F(x,y)/\partial x\} = -i\omega_x\Phi(\omega_x,\omega_y),$$

(7.18а)

$$O_\Phi\{\partial F(x,y)/\partial y\} = -i\omega_y\Phi(\omega_x,\omega_y).$$

(7.18б)

Следовательно, спектр лапласиана равен

$$O_\Phi\{\partial^2 F(x,y)/\partial x^2 + \partial^2 F(x,y)/\partial y^2\} = -(\omega_x^2+\omega_y^2)\Phi(\omega_x,\omega_y).$$

(7.19)

5.8. Анализ линейных систем с помощью преобразования Фурье

Теорема о преобразовании Фурье свертки (7.14) оказывается очень полезным средством при анализе линейных систем. Рассмотрим функцию $F(x,y)$, описывающую изображение на входе линейной системы с импульсным откликом $H(x,y)$. Изображение на выходе описывается функцией $G(x,y)$, получаемой в результате свёртки:

$$G(x,y) = \int_{-\infty}^{\infty}\int F(\alpha,\beta)H(x-\alpha,y-\beta)\,d\alpha d\beta.$$

(8.1)

Выполнив преобразование Фурье обеих частей этого равенства и поменяв порядок интегрирования в его правой части, получим

$$G(\omega_x,\omega_y)=\int_{-\infty}^{\infty}\int F(\alpha,\beta)\left[\int_{-\infty}^{\infty}\int H(x-\alpha,y-\beta)\exp(-i(\omega_x x+\omega_y y))dxdy\right]d\alpha d\beta. \tag{8.2}$$

Согласно теореме о сдвиге (7.13), внутренний интеграл равен произведению спектра функции $H(x,y)$ и экспоненциального множителя фазового сдвига. Поэтому

$$G(\omega_x,\omega_y)=\int_{-\infty}^{\infty}\int F(\alpha,\beta)\mathcal{H}(\omega_x,\omega_y)\exp\{-i(\omega_x\alpha+\omega_y\beta)\}d\alpha d\beta. \tag{8.3}$$

Выполнив преобразования Фурье, получим

$$G(\omega_x,\omega_y)=\mathcal{H}(\omega_x,\omega_y)F(\omega_x,\omega_y). \tag{8.4}$$

Наконец, обратное преобразование Фурье дает функцию, описывающую изображение на выходе:

$$G(x,y)=(1/4\pi^2)\int_{-\infty}^{\infty}\int \mathcal{H}(\omega_x,\omega_y)F(\omega_x,\omega_y)\exp(i(\omega_x x+\omega_y y))d\omega_x d\omega_y. \tag{8.5}$$

Выражения (8.1) и (8.5) представляют два альтернативных способа определения выходного изображения линейной пространственно-инвариантной системы. Выбор того или иного подхода зависит от решаемой задачи.

5.9. Обобщенные линейные системы

Ранее были введены понятия линейности и суперпозиции с тем, чтобы распространить понятие линейности на более широкий класс систем.

Рассмотрим две функции, описывающие изображения, $F_1(x,y)$ и $F_2(x,y)$, которые, взаимодействуя некоторым образом $(\Diamond)$, дают функцию $G(x,y)$:

$$G(x,y)=F_1(x,y)\Diamond F_2(x,y). \tag{9.1}$$

Пусть $O_G(\,)$ - оператор системы, преобразующей $G(x,y)$, который обладает следующими свойствами:

$$O_G(F_1(x,y)\Diamond F_2(x,y))=O_G(F_1(x,y))\vdots\Diamond O_G(F_2(x,y)) \tag{9.2а}$$

и

$$O_G(k:F(x,y))=k:O_G(F(x,y)). \tag{9.2б}$$

где k - постоянная, а двоеточие обозначает обобщённое умножение на постоянную. Показано, что если операция $\Diamond$ сводится к сложению векторов, а операция : - к умножению вектора на скаляр, то оператор $O_G(\cdot)$ может быть представлен в виде цепочки операторов, называемой **гомоморфным фильтром** (рис.9.1). Первый оператор $O_A(\cdot)$ превращает операции $\Diamond$ и : в сложение векторов и умножение вектора на скаляр:

$$O_A(F_1(x,y)\Diamond F_2(x,y))=O_A(F_1(x,y))+O_A(F_2(x,y)) \tag{9.3а}$$

и

$$O_A(k:F(x,y))=kO_A(F(x,y)). \tag{9.3б}$$

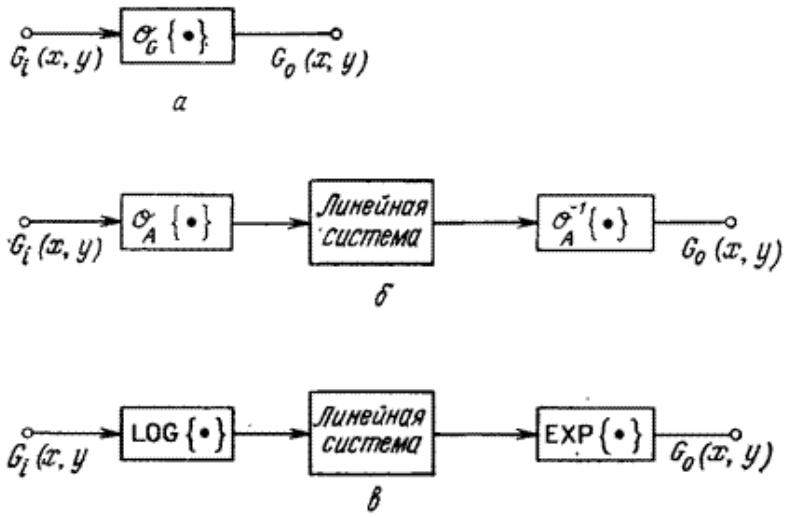


Рис.9.1. Обобщенные линейные системы: а — обобщенная система; б — представление обобщенной системы в виде гомоморфного фильтра; в - мультипликативный гомоморфный фильтр.

Вторая ступень гомоморфного фильтра – обычная линейная система.

Третья ступень – оператор O_A^{-1}, который является обратным относительно первого оператора, т.е.

$$O_A^{-1}\{O_A[F(x,y)]\} = F(x,y). \qquad (9.4)$$

Рис.9.1, в иллюстрирует частный случай гомоморфного фильтра для мультипликативной системы, в которой функция $G(x,y)$ получается в результате перемножения функций $F_1(x,y)$ и $F_2(x,y)$, т.е.

$$G(x,y) = F_1(x,y)\cdot F_2(x,y) = F_1(x,y)F_2(x,y). \qquad (9.5)$$

Прологарифмировав обе части равенства (9.5), получим сумму логарифмов функций $F_1(x,y)$ и $F_2(x,y)$:

$$\log\{G(x,y)\} = \log\{F_1(x,y)\} + \log\{F_2(x,y)\}. \qquad (9.6)$$

Функция $\log G(x,y)$ преобразуется некоторой линейной системой, а затем посредством экспоненциального преобразования возвращается в пространство исходных изображений. Операция обобщенного умножения вектора на скаляр определяется как возведение в степень

$$G(x,y) = k \cdot F(x,y) = [F(x,y)]^k \qquad (9.7)$$

Логарифмирование этого равенства дает

$$\log\{G(x,y)\} = k\log\{F(x,y)\}. \qquad (9.8)$$

5.10. Вероятностное описание непрерывных изображений

Часто бывает удобно рассматривать изображение как реализацию случайного процесса. Введем порождающую изображения непрерывную случайную функцию $F(x,y,t)$ трех переменных — пространственных координат x,y и времени t.

Случайный процесс $F(x,y,t)$ полностью описывается совместной плотностью вероятности

$$p(F_1,F_2,\ldots,F_J;x_1,y_1,t_1,x_2,y_2,t_2,\ldots,x_J,y_J,t_J)$$

для J значений функции $F_j(x_j,y_j,t_j)$ в точках отсчёта x_j,y_j,t_j. Совместные плотности вероятности высокого порядка для изображений обычно не известны, и их в общем случае трудно моделировать. Для плотности вероятности первого порядка $p(F,x,y,t)$ иногда удаётся подобрать удачную модель из

физических соображений или на основе измеренных гистограмм. Например, плотность вероятности первого порядка случайного шума в электронных преобразователях изображений хорошо моделируется гауссовой плотностью:

$$p(F;x,y,t) = [2\pi\sigma_p^2(x,y,t)]^{-1/2} \exp\left\{-\frac{[F(x,y,t)-\eta_p(x,y,t)]^2}{2\sigma_p^2(x,y,t)}\right\}, \quad (10.1)$$

где параметры $\eta_p(x,y,t)$ и $\sigma_p^2(x,y,t)$ есть среднее и дисперсия шума. Гауссова плотность может также с приемлемой точностью служить моделью плотности вероятности коэффициентов унитарных преобразований изображений. Плотность вероятности яркости должна быть односторонней, так как яркость принимает только положительные значения. В качестве моделей плотности вероятности яркости применяются плотность распределения вероятностей Рэлея

$$p(F;x,y,t) = \frac{F(x,y,t)}{\alpha^2} \exp\left\{-\frac{[F(x,y,t)]^2}{2\alpha^2}\right\}, \quad (10.2а)$$

плотность логарифмически нормального распределения

$$p(F;x,y,t) = [2\pi F^2(x,y,t)\sigma_p^2(x,y,t)]^{-1/2} \exp\left\{-\frac{[\log[F(x,y,t)]-\eta_p(x,y,t)]^2}{2\sigma_p^2(x,y,t)}\right\} \quad (10.2б)$$

и плотность экспоненциального распределения

$$p(F;x,y,t) = \alpha \exp(-\alpha[F(x,y,t)]). \quad (10.2в)$$

Эти плотности определены при $F \geq 0$, причем α - постоянная. Двусторонняя экспоненциальная, или Лапласова, плотность

$$p(F,\lambda,y,t) = (\alpha/2)\exp\{-\alpha|F(\lambda,y,t)|\}, \quad (10.3)$$

где α - постоянная, часто используется как модель плотности вероятности разностей отсчетов функции, описывающей изображение. Наконец, плотность равномерного распределения

$$p(F;x,y,t) = 1/2\pi, \quad -\pi \leq F \leq \pi, \quad (10.4)$$

Есть обычная модель для флуктуаций фазы случайного процесса. Для описания случайного процесса можно использовать так же условные плотности вероятности. Условная плотность вероятности значения функции $F(x,y,t)$ в точке (x_1,y_1,t_1) при заданном значении этой функции в точке (x_2,y_2,t_2) определяется как

$$p(F_1;x_1,y_1,t_1 \mid F_2;x_2,y_2,t_2) = \frac{p(F_1,F_2;x,y_1,t_1,x_1,y_2,t_2)}{p(F_2,x_2,y_2,t_2)}. \quad (10.5)$$

Аналогично определяются условные плотности более высокого порядка.

Другой способ описания случайного процесса состоит в вычислении средних по ансамблю. Первый момент, или среднее значение функции $F(x,y,t)$, равен

$$\eta_p(x,y,t) = E(F(x,y,t)) = \int_{-\infty}^{\infty} F(x,y,t)p(F;x,y,t)dF. \quad (10.6)$$

Второй момент, или автокорреляционная функция, определяется как

$$R(x_1,y_1,t_1;x_2,y_2,t_2) = E(F(x_1,y_1,t_1)F^*(x_2,y_2,t_2)) = \int_{-\infty}^{\infty}\int F(x_1,y_1,t_1)F^*(x_2,y_2,t_2)p(F_1,F_2;x_1,y_1,t_1,x_2,y_2,t_2)dF_1dF_2. \quad (10.7)$$

Автоковариационная функция изображения определяется как

$$K(x_1,y_1,t_1;x_2,y_2,t_2)=E\left\{[F(x_1,y_1,t_1)-\eta_F(x_1,y_1,t_1)][F^*(x_2,y_2,t_2)-\eta_F^*(x_2,y_2,t_2)]\right\} \quad (10.8а)$$

Автоковариационная и автокорреляционная функции связаны соотношением

$$K(x_1,y_1,t_1;x_2,y_2,t_2)=R(x_1,y_1,t_1;x_2,y_2,t_2)-\eta_F(x_1,y_1,t_1)\eta_F^*(x_2,y_2,t_2). \quad (10.8б)$$

наконец, дисперсия процесса $F(x,y,t)$ есть

$$\sigma_F^2(x,y,t)=K(x_1,y_1,t_1;x_2,y_2,t_2). \quad (10.9)$$

Случайный процесс, порождающий изображения, называется стационарным в строгом смысле, если его моменты не зависят от переноса начала координат в пространстве или времени. Процесс называется стационарным в широком смысле, если он имеет постоянную среднюю яркость, а его автокорреляционная функция зависит от разностей координат $x_1-x_2,\ y_1-y_2,\ t_1-t_2$, но не от самих координат. Для стационарного процесса $F(x_1,y_1,t_1)$

$$E(F(x_1,y_1,t_1))=\eta_F \quad (10.10а)$$

и

$$R(x_1,y_1,t_1;x_2,y_2,t_2)=R(x_1-x_2,y_1-y_2,t_1-t_2). \quad (10.10б)$$

Выражение для автокорреляционной функции можно записать в виде

$$R(\tau_x,\tau_y,\tau_t)=E\{F(x+\tau_x,y+\tau_y,t+\tau_t)F^*(x,y,t)\}. \quad (10.11)$$

Так как

$$R(-\tau_x,-\tau_y,-\tau_t)=R^*(\tau_x,\tau_y,\tau_t), \quad (10.12)$$

для действительной функции F автокорреляционная функция является действительной и четной. Энергетический спектр стационарного изображения по определению есть результат трехмерного преобразования Фурье его автокорреляционной функции:

$$W(\omega_x,\omega_y,\omega_t)=\int_{-\infty}^{\infty}\int_{-\infty}^{\infty}\int_{-\infty}^{\infty}R(\tau_x,\tau_y,\tau_t)\exp(-i[\omega_x\tau_x+\omega_y\tau_y+\omega_t\tau_t])d\tau_x d\tau_y d\tau_t. \quad (10.13)$$

Во многих изображающих системах пространственные и временные процессы формирования изображений разделяются. В этом случае стационарную автокорреляционную функцию можно записать как

$$R(\tau_x,\tau_y,\tau_t)=R_{xy}(\tau_x,\tau_y)R_t(\tau_t). \quad (10.14)$$

Часто для упрощения вычислений пространственную автокорреляционную функцию представляют в виде произведения автокорреляционных функций для каждой пространственной переменной:

$$R_{xy}(\tau_x,\tau_y)=R_x(\tau_x)R_y(\tau_y) \quad (10.15)$$

В изображениях объектов, созданных человеком, часто встречаются горизонтальные и вертикальные структуры, поэтому аппроксимация автокорреляционной функции произведением (10.15) оказывается вполне приемлемой. В изображениях естественных сцен обычно нет преобладающих направлений корреляции. Пространственная автокорреляционная функция таких изображений близка к функции с вращательной симметрией и не является поэтому разделимой.

Часто моделью изображения служат реализации двумерного Марковского процесса первого порядка. Автоковариационная функция такого процесса имеет вид

$$K_{x,y}(\tau_x,\tau_y)=K\exp\left(-\sqrt{\alpha_x^2\tau_x^2+\alpha_y^2\tau_y^2}\right), \quad (10.16)$$

где K, α_x и α_y — масштабные множители. Соответствующий энергетический спектр равен

$$W(\omega_x,\omega_y)=(1/\alpha_x\alpha_y)\left[\frac{2K}{1+[\omega_x^2/\alpha_x^2+\omega_y^2/\alpha_y^2]}\right] \quad (10.17)$$

Часто делается упрощающее предположение, что автоковариационная функция марковского процесса может быть представлена в виде

$$K_{x,y}(\tau_x,\tau_y)=K\exp(-\alpha_x|\tau_x|-\alpha_y|\tau_y|). \quad (10.18)$$

Энергетический спектр этого процесса есть

$$W(\omega_x,\omega_y)=(4\alpha_x\alpha_y K)/\left[(\alpha_x^2+\omega_x^2)(\alpha_y^2+\omega_y^2)\right]. \quad (10.19)$$

При детерминированном описании изображений были определены средние по пространству и времени. При статистическом описании также определяется среднее по ансамблю. Возникает вопрос: как связаны друг с другом пространственно-временные средние и средние по ансамблю? Ответ состоит в том, что для некоторых случайных процессов, называемых эргодическими, пространственно-временные средние и средние по ансамблю равны. Очень трудно доказать эргодичность случайного процесса в общем случае. Обычно достаточно определить эргодичность второго порядка, при которой моменты первого и второго порядка, полученные пространственно-временным усреднением, равны соответствующим моментам при усреднении по ансамблю.

5.11. Преобразование случайных изображений В изображающей системе

Часто известны плотность вероятности или моменты случайного поля, представляющего изображение, на входе некоторой системы и требуется определить соответствующие характеристики для изображения на выходе этой системы. Если передаточная функция системы представлена в виде алгебраического выражения, то можно найти плотность вероятности на выходе посредством преобразования плотности вероятности на входе. Пусть, например, функции, описывающие изображения на входе и выходе системы, связаны следующим образом:

$$G(x,y,t)=O_P(F(x,y,t)), \quad (11.1)$$

где $O_P(\cdot)$ - монотонный оператор, действующий на функцию $F(x,y,t)$. Тогда плотность вероятности на выходе есть

$$p(G,x,y,t)=\frac{p(F(x,y,t))}{\left|dO_P(F(x,y))/dF\right|}. \quad (11.2)$$

Преобразование вида (11.2) можно распространить на плотности вероятности высокого порядка, однако часто полученные соотношения оказываются слишком громоздкими.

Моменты случайного поля на выходе системы могут быть получены непосредственно, если известна его плотность вероятности, или в некоторых случаях косвенным путем, исходя из свойств оператора системы. Если, например, этот оператор является линейным, то среднее значение поля $G(x,y,t)$ есть

$$E(G(x,y,t))=E(O_L(F(x,y,t)))=O_L(E(F(x,y,t))). \quad (11.3)$$

Можно показать, что если оператор системы линеен, а случайное поле на входе системы стационарно в строгом смысле, то случайное поле на выходе также стационарно в строгом смысле. Если же поле на входе стационарно в широком смысле, то в том же смысле стационарно поле на выходе.

Рассмотрим линейную пространственно-инвариантную систему, изображение на выходе которой представляется интегралом свертки:

$$G(x,y,t) = \int_{-\infty}^{\infty}\int\int F(x-\alpha, y-\beta, t-\gamma)H(\alpha,\beta,\gamma)d\alpha d\beta d\gamma, \tag{11.4}$$

где $H(x,y,t)$ - импульсный отклик системы. Среднее значение $G(x,y,t)$ есть

$$E(G(x,y,t)) = \int_{-\infty}^{\infty}\int\int E(F(x-\alpha, y-\beta, t-\gamma))H(\alpha,\beta,\gamma)d\alpha d\beta d\gamma. \tag{11.5}$$

Если случайное поле на входе стационарно, то среднее значение η_F постоянно и может быть вынесено из под интеграла. Тогда

$$E(G(x,y,t)) = \eta_F\int_{-\infty}^{\infty}\int\int H(\alpha,\beta,\gamma)d\alpha d\beta d\gamma = \eta_F\mathcal{H}(0,0,0). \tag{11.6}$$

где $\mathcal{H}(0,0,0)$ есть частотная характеристика линейной системы в начале координат пространственно-временной частотной области.
Таким же способом легко показать, что автокорреляционные функции входного и выходного полей связаны соотношением

$$R_G(\tau_x,\tau_y,\tau_t) = R_F(\tau_x,\tau_y,\tau_t) * H(\tau_x,\tau_y,\tau_t) * H^*(-\tau_x,-\tau_y,\tau_t). \tag{11.7}$$

Выполнив преобразование Фурье обеих частей равенства (11.7) и используя теорему о преобразовании Фурье свертки, получим соотношение между энергетическими спектрами входного и выходного полей

$$W_G(\omega_x,\omega_y,\omega_t) = W_F(\omega_x,\omega_y,\omega_t)\mathcal{H}(\omega_x,\omega_y,\omega_t)\mathcal{H}^*(\omega_x,\omega_y,\omega_t), \tag{11.8а}$$

$$W_G(\omega_x,\omega_y,\omega_t) = W_F(\omega_x,\omega_y,\omega_t)\left|\mathcal{H}(\omega_x,\omega_y,\omega_t)\right|^2. \tag{11.8б}$$

Этот результат полезен при анализе действия шумов в изображающих системах.

5.12. Пространственные спектры изображений

При обработке изображений широко используется анализ спектров изображений. Спектр изображения получают прямым двумерным преобразованием Фурье функции, описывающей изображение:

$$F(\omega_x,\omega_y) = \int_{-\infty}^{\infty}\int_{-\infty}^{\infty} f(x,y)\exp(-i(\omega_x x + \omega_y y))dxdy, \tag{12.1}$$

где ω_x, ω_y - пространственные частоты; $i = \sqrt{-1}$, мнимая единица.
Функция $exp(-i(\omega_x x + \omega_y y))$ при фиксированных значениях пространственных частот описывает плоскую волну в плоскости изображения $f(x,y)$ (в соответствии с рисунком 12.1).
Формула (12.1) связывает вещественную функцию, описывающую яркость изображения $f(x,y)$ с комплексной функцией частоты - спектром изображения $F(\omega_x,\omega_y)$:

$$F(\omega_x,\omega_y)=\int_{-\infty}^{\infty}\int_{-\infty}^{\infty}f(x,y)\cos(\omega_x x+\omega_y y)dxdy+$$

$$+i\int_{-\infty}^{\infty}\int_{-\infty}^{\infty}(-f(x,y))\sin(\omega_x x+\omega_y y)dxdy=Re(\omega_x,\omega_y)+i\,Im(\omega_x,\omega_y)$$

(12.2)

где $Re(\omega_x, \omega_y)$ - реальная часть спектра; $Im(\omega_x, \omega_y)$ - мнимая часть спектра.

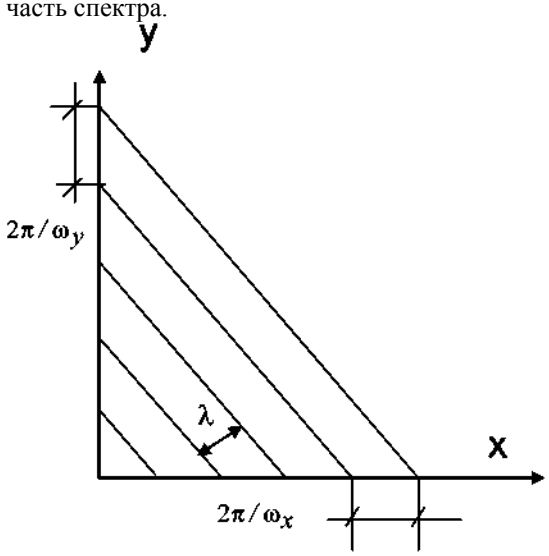


Рис. 12.1 Определение пространственных частот изображения.

Амплитуда и фаза спектра определяются по формулам (12.3) и (12.4) соответственно:

$$\mathrm{F}(\omega_x,\omega_y)=\sqrt{Re(\omega_x,\omega_y)^2+Im(\omega_x,\omega_y)^2},$$

$$\varphi(\omega_x,\omega_y)=\operatorname{arctg}(Im(\omega_x,\omega_y)/Re(\omega_x,\omega_y)).$$

(12.3)

Из (12.3)

$$F(\omega_x,\omega_y)=\mathrm{F}(\omega_x,\omega_y)\exp(i\varphi(\omega_x,\omega_y)). \quad (12.4)$$

Обратное преобразование Фурье позволяет восстановить изображение по его спектру:

$$f(x,y)=(1/4\pi^2)\int_{-\infty}^{\infty}\int F(\omega_x,\omega_y)\exp(i(\omega_x x+\omega_y y))d\omega_x d\omega_y .$$

(12.5)

5.13 Спектральные интенсивности изображений

Спектральная интенсивность изображения характеризует распределение энергии по пространственным частотам. Она определяется как квадрат модуля спектра изображения:

$$\mathrm{S}(\omega_x,\omega_y)=Re(\omega_x,\omega_y)^2+Im(\omega_x,\omega_y)^2=\mathrm{F}^2(\omega_x,\omega_y).$$

(13.1)

Для ее названия используются термины спектральная плотность и энергетический спектр.

Энергия изображения определяется как интеграл энергетического спектра по пространственным частотам. В соответствии с теоремой Парсеваля энергия изображения может быть вычислена в соответствии с (13.7):

$$\int_{-\infty}^{\infty}\int_{-\infty}^{\infty}f^2(x,y)dxdy=\int_{-\infty}^{\infty}\int_{-\infty}^{\infty}\left|F(\omega_x,\omega_y)\right|^2 d\omega_x d\omega_y .$$

(13.2)

5.14 Вероятностные модели изображений и функции автокорреляции

Вероятностные модели изображений широко используются для описания изображений. **Изображение в этом случае рассматривается как случайная функция пространственных координат *(x,y)* и времени *t.*** Случайный процесс называется *стационарным в широком смысле,* если он имеет постоянные значения математического ожидания и дисперсии, а его автокорреляционная функция зависит не от координат, а от их разностей (сдвига). Случайный процесс называется *стационарным в узком смысле,* если его *n*-мерная плотность распределения вероятностей инвариантна к сдвигу. В этом случае не зависят от времени и моменты более высокого порядка, в

частности, асимметрия и эксцесс. Случайный процесс описывается плотностью вероятности распределения яркости в изображении по пространственным координатам для некоторого фиксированного момента времени *t p(x,y)*.
В соответствии с определением математическое ожидание (среднее значение) стационарного процесса в широком смысле

$$Mf = \xi = \int_{-\infty}^{\infty}\int_{-\infty}^{\infty} f(x,y)\,p(x,y)\,dxdy = \text{const}. \quad (14.1)$$

Дисперсия

$$Df = \sigma^2 = E(f(x,y)-\xi)^2 = \int_{-\infty}^{\infty}\int_{-\infty}^{\infty} (f(x,y)-\xi)^2\,p(x,y)\,dxdy = \text{const}. \quad (14.2)$$

Функция автокорреляции вычисляется в соответствии с (14.3):

$$R(\tau_x,\tau_y) = \int_{-\infty}^{\infty}\int_{-\infty}^{\infty} f(x,y)\,f(x-\tau_x, y-\tau_y)\,dxdy, \quad (14.3)$$

где τ_x, τ_y задают сдвиги изображения по соответствующим осям координат.
Для действительной функции *f* автокорреляционная функция является действительной и четной.
Спектр двумерной автокорреляционной функции изображения (прямое преобразование Фурье автокорреляционной функции) равен энергетическому спектру изображения (спектральной плотности мощности) по определению:

$$S(\omega_x,\omega_y) = \int_{-\infty}^{\infty}\int_{-\infty}^{\infty} R(\tau_x,\tau_y)\exp(-i(\omega_x\tau_x+\omega_y\tau_y))\,d\tau_x d\tau_y. \quad (14.4)$$

Стационарный случайный процесс называется *эргодическим,* если любая его вероятностная характеристика может быть получена из одной реализации путем усреднения по времени. При этом среднее по времени равно среднему по ансамблю реализаций. Свойство эргодичности используется при оценке вероятностных характеристик изображений.

5.15 Критерии качества изображений

Качество изображения может определяться статистическими, спектральными, яркостными характеристиками изображения. В большинстве практических применений **качество рассматривается как мера близости двух изображений: реального и идеального или преобразованного и исходного. При таком подходе можно оценивать как субъективную степень похожести изображений, так и получать объективные оценки параметров сигналов изображения: моменты первого и второго порядка разностного сигнала сравниваемых изображений, такие параметры преобразования как отношение С/Ш, коэффициенты сжатия информации и другие.**
Субъективные критерии - это критерии визуального восприятия, оцениваемые в процессе экспертизы некоторой группой наблюдателей (экспертов). Наибольшее распространение получил **метод оценок**, при котором наблюдатель оценивает качество изображения в баллах по определенной шкале, считая, что идеальное изображение имеет максимальный балл. Этот метод позволяет оценить такие характеристики изображения как правильность цветопередачи, координатные искажения, чистоту переходов и др.
Для интерпретации полученных экспертных оценок разработаны методы их представления, например, построение кумулятивных кривых распределения оценок как функции от искажений. Средняя оценка определяется по формуле

$$g_{\text{ср}} = (1/N)\sum_{i=1}^{r} i n_i,$$

где N-общее число оценок, n_i- число оценок равных i баллам, r - количество видов разных оценок.
Нормализованные оценки *p* выражают относительное качество в диапазоне [0,1]. При пятибалльной системе, когда $g \in [1,5]$:

$$p = (g-1)/4,$$

а средняя оценка вычисляется в соответствии с формулой:

$$p_{\text{ср}} = (n_5 + 0{,}75n_4 + 0{,}5n_3 + 0{,}25n_2)/N.$$

Основные шкалы оценок при использовании метода сравнения приведены в таблице 1

Таблица 1

Основные шкалы субъективных оценок качества изображения

Норма-лизованная	Пятибалльная шкала		Семибалльная шкала ухудшения
	Шкала качества	Шкала ухудшения	
1	5 (отлично)	5 (незаметно)	1 (незаметно)
0,875	-	-	2 (едва заметно)
0,75	4 (хорошо)	4 (заметно, но не мешает)	3 (заметно, но лишь слегка ухудшает)
0,625	-	-	4 (ухудшает, но не мешает)
0,5	3 (удовлетворительно)	3 (заметно, немного мешает)	5 (несколько мешает)
0,25	2 (плохо)	2 (мешает)	6 (определенно мешает)
0	1 (очень плохо)	1 (сильно мешает)	7 (крайне мешает)

Единицей ухудшения качества телевизионных (ТВ) изображений является имп (от *impairment* -ухудшение, повреждение). Эта единица введена Проссером, Аллнаттом и Льюисом в 1964 г. и используется МККР (Международным консультативным комитетом по радиосвязи *(CCIR)).* Ухудшение обратно пропорционально нормализованной оценке качества и изменяется от ∞ до 0 при изменении *p* от 0 до 1 в соответствии с формулой:

$I = 1/p - 1$.

Достоинство методики оценки ухудшения состоит в том, что результирующая оценка ухудшения получается арифметическим суммированием оценок ухудшения, вызванных различными видами искажений сигналов изображения. Основываясь на психофизических свойствах наблюдателя, субъективные оценки позволяют характеризовать восприятие изображения. Интегральный критерий качества формируется по обобщенной формуле:

$$I_{\Sigma} = \sum_{i=1}^{M} I_i^{\nu},$$

где *M*-число параметров, по которым оценивается качество изображения; *v* - показатель степени.

Значение показателя степени принимают равным 1, но могут быть использованы, например, такие значения как 0,78 или 2. В настоящее время применяются и другие оценки качества изображений. При разработке аппаратных средств специального назначения большое

значение имеет оценка объективных характеристик качества преобразованного изображения.

Объективными критериями, используемыми при оценке качества изображений, являются критерии, позволяющие получить просто вычисляемую характеристику изображения разностного сигнала. К таким критериям относится, прежде всего, среднеквадратический критерий. По нему мерой различия двух изображений $f(x,y)$ и $f_{пр}(x,y)$ является среднеквадратическое значение разностного сигнала двух изображений. Для непрерывных изображений, заданных при $x \in [0,N]$ и $y \in [0,M]$, среднеквадратическое отклонение (СКО) вычисляется по формуле:

$$\sigma^2 = \int_0^M \int_0^N \left[f(x,y) - f_{пр}(x,y) \right]^2 dxdy. \quad (15.1)$$

В некоторых случаях используется критерий максимальной ошибки, который в отличие от (15.1), позволяет установить значение максимальной ошибки преобразования:

$$\varepsilon_{max} = \max_{(x,y)} \left| f(x,y) - f_{пр}(x,y) \right|. \quad (15.2)$$

Применяются и другие объективные критерии качества изображений . Существует определенное разногласие в оценках качества, даваемых человеческим глазом (субъективных), и объективных, полученных в виде количественных показателей. Глаз является совершенным изобретением природы, с ним не могут соревноваться достаточно примитивные объективные оценки типа СКО, пикового отношения сигнал/шум (ПСШ) и др. Поэтому некоторые результаты, рассматриваемые с точки зрения объективных оценок как одинаковые, визуально могут восприниматься различно. Однако объективные критерии используются при компьютерной обработке изображений в системах ИИ с автоматическим принятием решений.

Функционирование автоматических компьютерных систем ИИ полностью подчинено математическим критериям, и качество их работы оценивается только объективными показателями. Понятно, что и качество изображений, используемых в этих системах, также должно оцениваться только по объективным критериям.

6. Моделирование изображений случайными полями

Для эффективного решения различных задач обработки И необходима их математическая постановка, которая прежде всего включает в себя математическое описание, т. е. модель И как объекта исследования.

6.1. Случайные поля

Наиболее распространенными являются системы ИИ, включающие в себя пространственные системы датчиков и цифровую вычислительную технику. Поэтому мы будем в основном рассматривать МИ с дискретными пространственными и временными переменными. Не ограничивая общности, будем считать, что МИ заданы на многомерных прямоугольных сетках с единичным шагом. На рис. 1.1,а и 1.1,б изображены двумерная и трехмерная сетки. В общем случае И задано в узлах n-мерной сетки

$$\Omega = \{\bar{j} = (j_1,\dots,j_n): j_k = \overline{1,M_k}, k = \overline{1,n}\}.$$

В зависимости от физической природы значения И могут быть скалярными (например, яркость монохроматического изображения), векторными (поле скоростей, цветные изображения, поле смещений) и более сложнозначными (например, матричными). Если обозначить через $x_{\bar{j}}$ значение И в узле (пикселе) $\bar{j}$, то И есть совокупность этих значений на сетке: $X = \{x_{\bar{j}}: \bar{j} \in \Omega\}$.

Если данные представляют собой временную последовательность И, то иногда удобно считать эту последовательность одним И, увеличив размерность сетки на единицу. Например, последовательность из плоских И (рис. 1.1,а) можно рассматривать как одно трехмерное И (рис. 1.1,б).

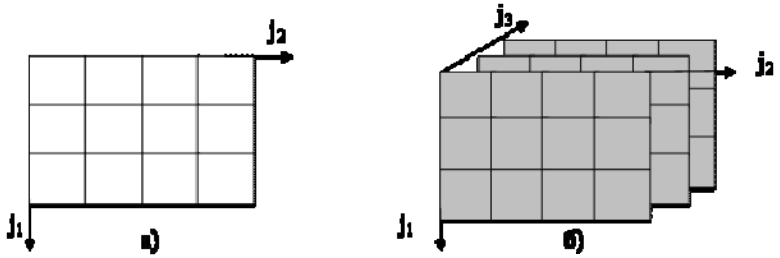


Рис. 1.1.

Если требуется временную переменную выделить особо, то будем ее записывать сверху: $X = \{x_{\bar{j}}^i: \bar{j} \in \Omega, i \in I\}$. Это И задано на прямом произведении $\Omega \times I$ сеток Ω и I, где I – множество значений временного индекса. Сечение $x^i = \{x_{\bar{j}}^i: \bar{j} \in \Omega\}$, т.е. совокупность отсчетов И при фиксированном значении временного индекса *i*, называется *i*-м кадром И. Каждый кадр задан на сетке Ω. Например, на рис. 1.1,б изображено три двухмерных кадра.

Таким образом, **МИ можно рассматривать как некоторую функцию, определенную на многомерной сетке**. Значение элементов И невозможно точно предсказать заранее (иначе система наблюдения была бы не нужна), поэтому естественно рассматривать эти значения как случайные величины (СВ), применяя аппарат теории вероятностей и математической статистики. **Итак, приходим к основной модели МИ – системе СВ, заданных на многомерной сетке. Такие системы называются дискретными случайными полями (СП) или случайными функциями нескольких переменных.**

Для описания СП, как и любой другой системы СВ, можно задать совместную функцию распределения вероятностей (ФР) его элементов $F(u_{\bar{j}}: \bar{j} \in \Omega) = P(x_{\bar{j}} < u_{\bar{j}}: \bar{j} \in \Omega)$ или совместную плотность распределения вероятностей (ПРВ) $\omega(u_{\bar{j}}: \bar{j} \in \Omega)$. Однако И обычно состоит из очень большого количества элементов (тысячи и

миллионы), поэтому ФР (или ПРВ) при таком количестве переменных становится необозримой и требуются другие, менее громоздкие методы описания СП.

6.2. Авторегрессионные модели случайных последовательностей

Рассмотрим сначала одномерное И, т. е. случайный процесс X с дискретным временем, который можно считать случайной функцией $x(i,\xi)$ или последовательностью случайных величин (СВ) $x^i = x^i(\xi)$. Здесь номер отсчета i=0, 1, 2, … часто может интерпретироваться как дискретное время. Параметр ξ в соответствии с некоторым вероятностным законом принимает в каждом проводимом опыте конкретное значение ξ_0, от которого зависит вид функции $x(i,\xi_0)$, называемой **реализацией процесса** в данном опыте.

Пусть, например, $x(i,\xi) = x(i,(A,w)) = A\sin(wi)$, где $\xi = (A,w)$ – система двух СВ. Тогда реализациями процесса будут синусоиды со случайной амплитудой и частотой. На практике часто приходится иметь дело с процессами, реализации которых имеют более сложный вид, поэтому и функция $x(i,\xi)$ должна иметь достаточно сложную структуру и в то же время допускать эффективное решение задач анализа, синтеза и имитации. Удачными в этом отношении являются авторегрессионные модели, в которых используются рекурсивные функции.

Рассмотрим простейшую **авторегрессионную модель** случайного процесса $X = \{x_0, x_1, x_2, \ldots\}$. Пусть последовательность СВ, составляющих данный процесс, удовлетворяет стохастическому уравнению

$$x^i = \varphi(x^{i-1}, \xi^i), i = 1,2,\ldots \qquad (2.1)$$

с начальным условием $x^0 = \varphi^0(\xi^0)$, где φ и φ^0 – некоторые функции; $\xi = (\xi^0, \xi^1, \xi^2, \ldots)$ – заданная последовательность независимых СВ, называемая **порождающей или возмущающей последовательностью**.

Характерной особенностью модели (2.1) является ее **каузальность**, т.е. возможность вычисления значения x^i как функции предыдущего значения x^{i-1} и случайного числа ξ^i. Свойство каузальности позволяет не только эффективно решать задачи имитации, но и дает возможность построения мощных рекуррентных алгоритмов оценивания типа фильтра Калмана.

Заметим, что условная ПРВ величин $x^{i+1}, x^{i+2}, x^{i+3}, \ldots$, описываемых уравнением (2.1), при известном значении x^i не зависит от $x^0, x^1, \ldots, x^{i-1}$. Таким образом, данная последовательность является **марковской**: ее «будущее» $\Gamma^+ = \{x^k : k > i\}$ в указанном смысле условно независимо от «прошлого» $\Gamma^- = \{x^k : k < i\}$ при известном «настоящем» $\Gamma = \{x^i\}$. Если сменить направление дискретного времени на обратное, то полученная последовательность $\ldots, x^2, x^1, x^0$ также будет марковской. Поэтому марковское свойство может быть сформулировано в симметричной форме: последовательность является марковской, если ПРВ величин, составляющих Γ^- и Γ^+, условно независимы при известном Γ.

В качестве возмущающей последовательности чаще всего используется последовательность стандартных независимых гауссовских СВ. В случае линейности функции φ процесс X также

будет гауссовским, а при соответствующем выборе нелинейных функций и негауссовских возмущений можно получить широкий класс негауссовских процессов. В целях дальнейшего расширения класса представимых случайных процессов с помощью авторегрессионных моделей можно в (2.1) взять уравнение $x^t = \varphi^t(x^{t-1}, \xi^t)$, что дает возможность получать **неоднородные** марковские процессы. Стохастическое уравнение $x^t = \varphi^t(x^{t-m}, x^{t-m+1}, \ldots, x^{t-2}, x^{t-1}, \xi^t)$ при соответствующих начальных условиях порождает **марковский процесс m-го порядка**,

у которого прошлое $\Gamma^- = \{x^k : k \le i - m\}$ и будущее $\Gamma^+ = \{x^k : k > i\}$ условно независимы при известном настоящем $\Gamma = \{x^{i-m+1}, \ldots, x^i\}$, состоящем из m последовательных значений процесса.

Задача анализа авторегрессионной модели состоит в нахождении закон распределения СВ x_i. Она особых затруднений не вызывает, так как все x_i являются известными функциями СВ ξ^i. Более сложна задача **синтеза**, состоящая в нахождении функций φ^t авторегрессионной модели, порождающей процесс с заданными ПРВ.

Рассмотрим в качестве примера простейшую **линейную** авторегрессионную модель

$$x^0 = \sigma\xi^0, x^t = \rho x^{t-1} + \sigma\sqrt{1-\rho^2}\xi^t, t = 1,2,\ldots \quad (2.2)$$

где ξ^i – одинаково распределенные независимые стандартные СВ. Эта модель порождает стационарную марковскую последовательность

x_i с нулевым средним, дисперсией σ^2 и ковариационной функцией (КФ) $V_x(k) = M[x^t x^{t+k}] = \sigma^2 \rho^{|k|}$. Параметр ρ равен коэффициенту корреляции между соседними элементами порождаемого процесса, а σ равно среднеквадратическому отклонению (СКО) процесса.

На рис. 2.1 представлены типичные графики реализаций такого процесса при различных значениях параметра ρ, входящего в модель (2..2). Во всех случаях параметр σ, влияющий только на масштаб по оси ординат, выбран равным единице. Из этих рисунков видно, что при ρ, близких к единице, процесс становится более гладким; при малых ρ, напротив, значения процесса слабо зависимы между собой; при отрицательных ρ корреляция между соседними значениями процесса отрицательна, поэтому он часто меняет знак.

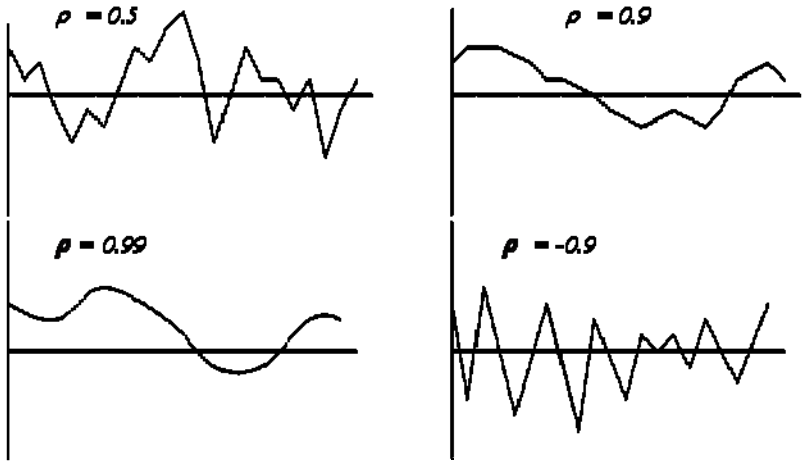


Рис. 2.1.

При гауссовской возмущающей последовательности ξ_i порождаемый процесс X в силу линейности модели будет гауссовским. В случае негауссовских возмущений X не будет гауссовским, но при |ρ|, близких к единице, нормализуется. Это следует из центральной предельной теоремы и представления модели (2.2) в виде **взвешенных сумм возмущений**:

$$x^i = \sigma\sqrt{1-\rho^2}\,(\xi^i + \rho\xi^{i-1} + \rho^2\xi^{i-2} + \ldots + \rho^{i-1}\xi^1) + \sigma\rho^i\xi^0.$$

Более сложные авторегрессионные модели

$$x^i = \sum_{j=1}^{m} \rho_j x^{i-j} + \beta\xi^i, i = m, m+1, \ldots \qquad (2.3)$$

с соответствующими начальными условиями на ПРВ значений первых m членов $x^0, x^1, \ldots, x^{m-1}$ определяют **марковские последовательности m-го порядка**.

Найдем КФ $V_x(i,j) = M[x^i x^j]$ порождаемой последовательности. Умножая (2.3) на x_0 и находя математические ожидания, получим рекуррентное соотношение для КФ:

$$V_x(0,i) = \sum_{j=1}^{m} \rho_j V_x(0, i-j), i = m, m+1, \ldots, \qquad (2.4)$$

которое вместе со значениями $V_x(0,i)$, $i<m$, определяемыми из начальных условий, дает возможность последовательно вычислять значения этой функции. Общее решение уравнения (2.4) имеет вид

$$V_x(0,k) = \sum_{u=1}^{n}\sum_{j=0}^{q_u-1} b_{u,j} k^j z_u^k, \qquad (2.5)$$

где $b_{u,j}$ – константы; z_u – корни **характеристического уравнения**

$$z^m - \sum_{j=1}^{m} \rho_j z^{s-j} = 0, \qquad (2.6)$$

q_u – кратность корня z_u. Каждому простому действительному корню

$z_u = \alpha_u$ характеристического уравнения (2.6) в (2.5) соответствует компонента α_u^k; простой паре комплексных корней $z_u = \alpha_u \pm ib_u$ – две компоненты $\alpha_u^k \cos(\beta_u^k)$ и $\alpha_u^k \sin(\beta_u^k)$; если z_u имеет кратность $q_u>1$, то добавляются произведения указанных компонент на $k, k^2, \ldots, k^{q_u-1}$. Полагая в (2.5) $k = 0,1,2,\ldots,m-1$ и используя начальные условия, получим m линейных уравнений для определения m констант $b_{u,j}$.

Аналогичным образом определяется КФ $V_x(i,i+k)$ при i>0, также имеющая вид (2.5) с константами, вообще говоря, зависящими от i.

Если все корни уравнения (2.6) по модулю меньше единицы, то $V_x(i,i+k)\to 0$ при $k\to\infty$, следовательно, x_i и x_{i+k} становятся независимыми, когда время k между ними стремится к бесконечности. Последовательность в этом случае приближается к **стационарной**: $V_x(i,i\pm k)\to V_x(k)$ при $i\to\infty$. Для нахождения этих предельных значений КФ умножим (2.3) на x_{i+k} и перейдем к пределу математических ожиданий при $i\to\infty$:

$$V_x(|k|) = \sum_{j=1}^{m} \rho_j V_x(|k-j|) + \beta^2\delta(k), \qquad (2.7)$$

где δ(k) – символ Кронекера. Полагая в (2.7) k = 0,1,2,…,m, получим m+1 линейных уравнений, из которых находятся m+1 начальных значений $V_x(0)$, $V_x(1)$, …, $V_x(m)$, которые можно использовать для нахождения $V_x(k)$ в форме (2.5), где в правой части вместо k нужно взять |k|.

Для того чтобы последовательность xi была стационарной с самого начала, необходимо и достаточно задать распределения начальных членов $x^0, x^1, \ldots, x^{m-1}$ так, чтобы их КФ удовлетворяла уравнению (2.7).

6.3. Авторегрессионные модели случайных полей

По своему строению СП значительно сложнее случайных процессов. Во-первых, реализации случайных полей являются функциями нескольких переменных, теория которых принципиально сложнее теории функций одной переменной. Во-вторых, значительно усложняется понятие марковости.

Случайный процесс можно представить развивающимся во времени, математическим выражением такого развития и является модель (2.1). Для марковских последовательностей временной интервал может быть разбит любой точкой *i* на условно независимые прошлое $\Gamma^- = \{x^k : k < i\}$ и будущее $\Gamma^+ = \{x^k : k > i\}$. Однако СП определено на *n*-мерной области W, для геометрического разбиения которого на две части Γ^- и Γ^+ требуется, по меньшей мере, (n-1)-мерная область Γ. **Свойство марковости случайного поля** состоит в том, что для любого множества Γ (из некоторого класса множеств) СВ, входящие в Γ^-, условно независимы от СВ, входящих в Γ^+, при известных значениях Γ. Назвать Γ^-, Γ и Γ^+ прошлым, настоящим и будущим можно весьма условно. Тем не менее марковское свойство позволяет представить случайное поле также формирующимся во времени от Γ^- через Γ к Γ^+, при этом

Γ с течением времени перемещается по W. Например, если в качестве Γ брать строки двумерной сетки W, то поле X можно представить формирующимся построчно.

Дальнейшее развитие этой идеи позволяет обобщить авторегрессионные модели случайных последовательностей на СП.

Если порядок формирования последовательности $x^0, x^1, x^2, \ldots$ обычно соответствует наблюдаемым во времени значениям, то порядок формирования поля $X = \{x_j : \bar{j} \in \Omega\}$ требует дополнительного определения. Для этого нужно линейно упорядочить узлы сетки Ω, тогда про любые два элемента поля можно сказать, что один из них предшествует другому. Если x_i предшествует x_j, то будем отмечать это как $(i) < (j)$, т. е. номер элемента x_i меньше номера x_j при данной развертке. Существует множество вариантов такого упорядочения. В двумерном случае чаще всего применяются пилообразная и треугольная развертки, показанные соответственно на рис. 3.1,а и 3.1,б.

Рис.3.1.

В результате развертки поле преобразуется в **случайную последовательность**. Предположим, что она является марковской порядка s, т. е. условная ПРВ любого x_i относительно всех предшествующих ему элементов зависит только от некоторого конечного отрезка $\Gamma_i = \{x_j : (i) - s \le (j) < (i)\}$. Множество Γ_i называется **глобальным состоянием**. В двумерном случае оно при

пилообразной (и треугольной) развертке включает в себя несколько последних строк и показано на рис.3.2. Следовательно, можно представить x_i в каузальном виде как функцию элементов глобального состояния и возмущения ξ_i:

$$x_i = \varphi_i(x_j : \bar{j} \in \Gamma_i; \xi_i) \quad (3.1)$$

Полученное выражение представляет **авторегрессионную модель случайного поля**. Однако использовать (3.1) для представления полей на сетках больших размеров трудно, а для бесконечных сеток – невозможно ввиду большого или даже бесконечного числа аргументов функций φ_i.

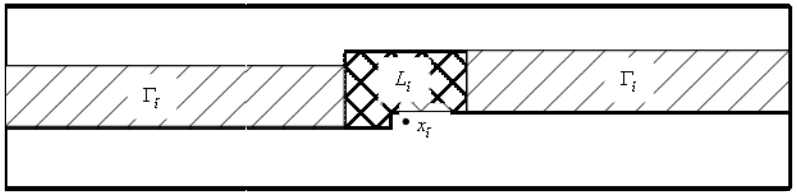


Рис.3.2.

Преодолеть эту трудность позволяет то обстоятельство, что ПРВ x_i часто зависит не от всего глобального состояния Γ_i, а только от некоторой его части L_i, называющейся **локальным состоянием** и включающей в себя только достаточно близкие к x_i элементы поля, не упреждающие x_i относительно данной развертки. На рис. 3.2 область, соответствующая локальному состоянию L_i, обозначена двойной штриховкой.

В результате поле X может быть представлено авторегрессионной моделью

$$x_i = \varphi_i(x_j : \bar{j} \in L_i; \xi_i), \quad (3.2)$$

которая во многих случаях может быть приемлема для решения прикладных задач. Конечно, может оказаться, что даже область локального состояния L_i слишком велика, и возникают значительные технические трудности при имитации или обработке полей. В таких ситуациях можно L_i уменьшить до приемлемых размеров, используя полученную модель (3.2) как некоторое приближение к реальным физическим объектам.

Линейные модели случайных полей

Рассмотрим **линейную гауссовскую авторегрессионную модель**

$$x_i = \sum_{j \in D} \alpha_j x_{i+j} + \beta \xi_i, \quad (3.3)$$

где применена многомерная развертка сетки W. Здесь α_j – весовые коэффициенты;
$\bar{i} + \bar{j} = (i_1, i_2, \ldots, i_n) + (j_1, j_2, \ldots, j_n) = (i_1 + j_1, \ldots, i_n + j_n) \{(\bar{i} + \bar{j}) : j \in D\} = L_i$ – локальное состояние; $\{\xi_i : \bar{i} \in \Omega\}$ – система стандартных гауссовских СВ. Одним из первых подобную модель применительно к оценке плоских изображений исследовал Хабиби:

$$x_{i_1 i_2} = \rho_1 x_{i_1-1,i_2} + \rho_2 x_{i_1,i_2-1} - \rho_1\rho_2 x_{i_1-1,i_2-1} + \sigma\sqrt{(1-\rho_1^2)(1-\rho_2^2)}\,\xi_{i_1 i_2} \quad (3.4)$$

.

Схема вычислений для этой модели представлена на рис.3.3,а.

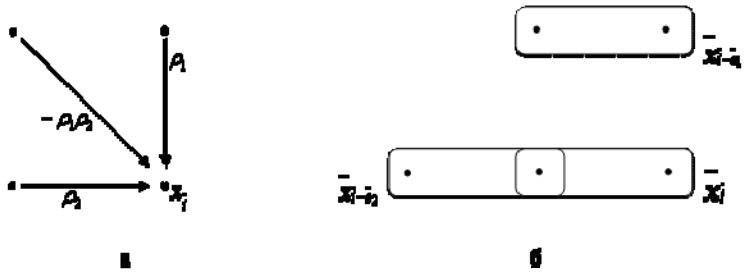


Рис. 3.3

Порождаемое поле имеет **множительную экспоненциальную** КФ:

$$V_x(i_1,i_2) = M[x_{i_1 i_2} x_{00}] = \sigma^2 \rho_1^{|i_1|} \rho_2^{|i_2|} \quad (3.5)$$

Обобщением модели (3.4) на многомерный случай является

$$x_{i_1,\dots,i_n} = \sigma \prod_{k=1}^{n} (1-\rho_k^2)^{1/2} \xi_{i_1,\dots,i_n} - \sum_{j_1,\dots,j_n=0}^{1} \prod_{k=1}^{n} (-\rho_k)^{j_k} x_{i_1-j_1,\dots,i_n-j_n} \quad (3.6)$$

Порождаемое поле также имеет множительную КФ

$$V_x(i_1,\dots,i_n) = \sigma^2 \prod_{k=1}^{n} \rho_k^{|i_k|} \quad (3.7)$$

На основе модели (3.4) разработано большое количество алгоритмов фильтрации случайных полей. Однако она, как и ее многомерный вариант (3.6), имеет существенный недостаток – множительность

(факторизуемость) КФ. В двумерном случае элементы поля, одинаково коррелированные с элементом $x_{i_1 i_2}$, расположены на ромбе с центром в (i_1,i_2), а в многомерном случае – на ромбоиде, хотя более естественными сечениями КФ для реальных полей были бы эллипс и эллипсоид. Для частичного скругления сечений КФ можно отказаться от частного вида весовых коэффициентов (3.6), как это сделано в п.6.4, а также расширить область локальных состояний. Однако такое расширение приводит к резкому увеличению числа слагаемых в (3.3).

Простейшее авторегрессионное уравнение, порождающее n-мерное поле X, не распадающееся на независимые поля меньшей размерности, имеет вид

$$x_i = \sum_{k=1}^{n} \alpha_k x_{i-\bar{e}_k} + \beta \xi_i, \quad (3.8)$$

где $\bar{e}_k = (0,\dots 0,1,0,\dots,0)$ – единичный вектор k-й координатной оси. Любая модель (3.3) может быть приведена к модели типа (3.8), содержащей минимально возможное число слагаемых. Для этого воспользуемся **векторными авторегрессионными моделями**, которые в линейном случае описываются уравнением

$$\bar{x}_i = \sum_{j \in D} A_j \bar{x}_{i+j} + B \bar{\xi}_i, \quad (3.9)$$

где $\bar{x}_i$ – значение векторного поля в узле i; A_j, B – квадратные матрицы; $\{\bar{\xi}_i\}$ – порождающее стандартное векторное поле независимых векторов с независимыми компонентами.

Действительно, рассмотрим, к примеру, модель (3.4) и представим ее в виде (3.9). Введем для этого векторы $\bar{x}_i = \bar{x}_{ij} = (x_{ij}, x_{i,j-1})^T$ и

$\bar{\xi}_i = (\xi_{ij}, \eta_{ij})^T$, $\beta = \sigma\sqrt{(1-\rho_1^2)(1-\rho_2^2)}$, тогда $\bar{x}_{i-\bar{e}_1} = (x_{i-1,j}; x_{i-1,j-1})^T$, $\bar{x}_{i-\bar{e}_2} = (x_{i,j-1}, x_{i,j-2})^T$ и уравнение (3.4) будет эквивалентно первой компоненте векторного уравнения

$$\begin{pmatrix} x_{ij} \\ x_{i,j-1} \end{pmatrix} = \begin{pmatrix} \rho_1 & -\rho_1\rho_2 \\ 0 & 0 \end{pmatrix} \begin{pmatrix} x_{i-1,j} \\ x_{i-1,j-1} \end{pmatrix} + \begin{pmatrix} \rho_2 & 0 \\ 1 & 0 \end{pmatrix} \begin{pmatrix} x_{i,j-1} \\ x_{i,j-2} \end{pmatrix} + \begin{pmatrix} \beta & 0 \\ 0 & 0 \end{pmatrix} \begin{pmatrix} \xi_{ij} \\ \eta_{ij} \end{pmatrix} \quad (3.10)$$

или при очевидных обозначениях

$$\bar{x}_i = A_1 \bar{x}_{i-\bar{e}_1} + A_2 \bar{x}_{i-\bar{e}_2} + B\bar{\xi}_i, \quad (3.11)$$

т. е. будет минимальной векторной моделью вида (3.9). На рис.3.3,б показаны элементы поля, входящие в векторы модели (3.10).

6.4. Анализ авторегрессионных моделей случайных полей

Установим связь КФ с параметрами скалярных и векторных авторегрессионных моделей СП. Для этого рассмотрим сначала гауссовскую скалярную модель (2.10), определяющую на бесконечной сетке Ω скалярное гауссовское поле X, которое полностью характеризуется своей КФ $V_X(\bar{l}) = M[x_i \cdot x_0] = M[x_{-i} \cdot x_0]$.

Введем **оператор сдвига** $\bar{z}^{-\bar{j}} x_i = x_{i+\bar{j}}$, где $\bar{z}^{-\bar{j}} = (z_1, \dots z_n)^{(j_1,\dots j_n)} = z_1^{j_1}, \dots z_n^{j_n}$, тогда каждой функции $g(\bar{z})$ n комплексных переменных, разложимой в конечную сумму

$\sum_j g_j \bar{z}^{-\bar{j}}$ или многомерный ряд Лорана по этим переменным, можно поставить в соответствие оператор, определяемый равенством $g(\bar{z})x_i = (\sum_j g_j \bar{z}^{-\bar{j}})x_i = \sum g_j x_{i+j}$. Введенные операторы линейны и арифметическим операциями над функциями соответствуют те же операции над соответствующими им операторами. В частности, $g(\bar{z})(h(\bar{z})x_i) = (g(\bar{z})(h(\bar{z}))x_i$, поэтому $g^{-1}(\bar{z}) = 1/g(\bar{z})$ определяет оператор, обратный к оператору, определяемому функцией $g(\bar{z})$. Уравнение (2.10) теперь можно записать в виде

$$x_i = (\sum_{j \in D} \alpha_j \bar{z}^{-\bar{j}})x_i + \beta\xi_i,$$

или

$$(1 - \sum_{j \in D} \alpha_j \bar{z}^{-\bar{j}})x_i = \beta\xi_i. \quad (4.1)$$

Применяя обратный оператор, получаем уравнение

$$x_i = \beta(1 - \sum_{j \in D} \alpha_j \bar{z}^{-\bar{j}})^{-1}\xi_i, \quad (4.2)$$

выражающее поле X через элементы возмущающего поля. Разлагая $\beta(1 - \sum_{j \in D} \alpha_j \bar{z}^{-\bar{j}})^{-1} = g(\bar{z})$ в ряд $\sum_{j \in \Omega} g_j \bar{z}^{-\bar{j}}$, получаем представление поля X в виде взвешенных сумм элементов возмущающего поля:

$$x_{\bar{i}} = \sum_{\bar{j}\in\Omega} g_{\bar{j}}\varepsilon_{\bar{i}+\bar{j}} \quad . \qquad (4.3)$$

Используя **спектральное представление** (4.3) и учитывая, что $M[\varepsilon_{\bar{i}}\varepsilon_{\bar{j}}] = \sigma^2(\bar{i}-\bar{j})$, находим КФ:

$$V_x(\bar{i}) = M[x_{\bar{i}}x_{\bar{0}}] = M[(\sum_{\bar{j}\in\Omega} g_{\bar{j}}\varepsilon_{\bar{i}+\bar{j}})(\sum_{\bar{k}\in\Omega} g_{\bar{k}}\varepsilon_{\bar{k}})] = \sum_{\bar{k}-\bar{j}=\bar{i}} g_{\bar{j}}g_{\bar{k}} \quad .$$

Заметим, что полученное выражение равно коэффициенту при $\bar{z}^{-\bar{i}}$ в разложении произведения $g(\bar{z})g^*(\bar{z}) = g(\bar{z})g(\bar{z}^{-1})$ в ряд, следовательно,

$$V_x(\bar{i}) = \sum_{\bar{i}\in\Omega} V_x(\bar{i})\bar{z}^{-\bar{i}} = \beta^2(1-\sum_{\bar{j}\in D}\alpha_{\bar{j}}\bar{z}^{-\bar{j}})^{-1}(1-\sum_{\bar{j}\in D}\alpha_{\bar{j}}\bar{z}^{-\bar{j}})^{-1} \quad . \qquad (4.4)$$

Функция n комплексных переменных $V_x(\bar{z})$ называется **энергетическим спектром** поля X, коэффициенты его разложения в n-мерный ряд Лорана равны соответствующим значениям КФ и могут быть найдены с помощью n-кратного интеграла

$$V_x(\bar{i}) = \frac{1}{(2\pi)^n}\int_{C_n} V_x(\bar{z})\bar{z}^{-\bar{i}-\bar{1}}d\bar{z} \quad , \qquad (4.5)$$

где $d(\bar{z}) = dz_1 \ldots dz_n$; $\bar{1} = (1,1,\ldots,1)$ и область интегрирования – единичная полиокружность (прямое произведение единичных

окружностей) $C_n = \{|z_1| = 1, \ldots, |z_n| = 1\}$. Из (4.4) и (4.5) получаем окончательное выражение КФ:

$$V_x(\bar{i}) = \frac{\beta^2}{(2\pi)^n}\int_{C_n}\frac{\bar{z}^{-\bar{j}-\bar{1}}}{(1-\sum_{\bar{j}\in D}\alpha_{\bar{j}}\bar{z}^{-\bar{j}})(1-\sum_{\bar{j}\in D}\alpha_{\bar{j}}\bar{z}^{-\bar{j}})}d\bar{z} \quad . \qquad (4.6)$$

Интеграл в этом выражении может быть найден с помощью вычетов или численными методами.

Вычисления значительно упрощаются, если $g(\bar{z})$ факторизуется, т. е. представляется в виде произведения n функций одного переменного: $g(\bar{z}) = \beta\, g_1(z_1)g_2(z_2)\ldots g_n(z_n)$. Тогда интеграл (4.6) превращается в произведение n однократных интегралов. Если возмущающее поле $\{\varepsilon_{\bar{i}}\}$ состоит из коррелированных величин, то в правую часть (4.4) и в числитель интеграла (4.6) добавится энергетический спектр $V_\varepsilon(z) \neq const$, так как из (4.3) следует соотношение

$$V_x(\bar{z}) = g(\bar{z})V_\varepsilon(\bar{z})g^*(\bar{z}) \quad . \qquad (4.7)$$

Перейдем теперь к рассмотрению **векторного стационарного СП** $X = \{\bar{x}_{\bar{i}} : \bar{i}\in\Omega\}$, порождаемого на бесконечной n-мерной сетке векторной авторегрессионной моделью (3.9). Поле X имеет **матричнозначные** КФ $V_X(\bar{i}) = M[\bar{x}_{\bar{i}}\bar{x}_{\bar{0}}^T] = V_X^T(-\bar{i})$ и энергетический спектр $V_X(\bar{z}) = \sum_{\bar{i}\in\Omega} V_X(\bar{i})\bar{z}^{-\bar{i}}$, являющийся суммой

многомерного ряда с матричными коэффициентами. Как и в скалярном случае, с помощью оператора сдвига $\bar{z}^{-j}\bar{x}_i = \bar{x}_{i+j}$ получим

$$\bar{x}_i = \sum_{j\in\Omega} g_j \bar{\xi}_{i-j}$$

представление поля в виде взвешенных сумм .
Аналогично находится связь между энергетическими спектрами

$$V_x(\bar{z}) = g(\bar{z}) V_\xi(\bar{z}) g^*(\bar{z})$$

и интегральное представление КФ:

$$V_x(\bar{i}) = \frac{1}{(2\pi i)^n} \int_{C_n} g(\bar{z}) V_\xi(\bar{z}) g^*(\bar{z}) \bar{z}^{-\bar{i}-\bar{1}} d\bar{z} \quad , \qquad (4.8)$$

$g(\bar{z}) = (E - \sum_{j\in J} A_j \bar{z}^{-j})^{-1} B$

где ; E – единичная матрица;

$g^*(\bar{z}) = g^T(\bar{z}^{-1})$; g_j – матричные коэффициенты разложения $g(\bar{z})$ в n-мерный ряд. В формуле (4.8) каждый элемент подынтегральной матрицы интегрируется независимо от остальных ее элементов.

Если возмущающее поле состоит из независимых случайных векторов, то $V_\xi(\bar{z}) \equiv E$ и (4.8) принимает вид

$$V_x(\bar{i}) = \frac{1}{(2\pi i)^n} \int_{C_n} [E - \sum_{j\in J} \bar{z}^{-j} A_j]^{-1} B B^T [E - \sum_{j\in J} \bar{z}^{-j} A_j^T]^{-1} \bar{z}^{-\bar{i}-\bar{1}} d\bar{z} \quad . \qquad (4.9)$$

После транспонирования получим

$$V_x(-\bar{i}) = \frac{1}{(2\pi i)^n} \int_{C_n} [E - \sum_{j\in J} \bar{z}^{j} A_j]^{-1} B B^T [E - \sum_{j\in J} \bar{z}^{j} A_j^T]^{-1} \bar{z}^{\bar{i}-\bar{1}} d\bar{z} \quad .$$

Будем предполагать, что матрица $E - \sum_{j\in J} \bar{z}^{-j} A_j$ неособенная при $\bar{z} \in C_n$, тогда интегралы (4.9) существуют.

Трехточечная модель

Рассмотрим в качестве примера двумерную **трехточечную** авторегрессионную линейную гауссовскую модель

$$x_{m,n} = a x_{m-1,n} + b x_{m,n-1} - c x_{m-1,n-1} + \beta \xi_{m,n} \quad , \qquad (4.10)$$

порядок вычисления для которой показан на рис.4.1,а.

Рис. 4.1

Заменим ее эквивалентной векторной моделью

$$\bar{x}_{n} = \begin{pmatrix} x_{n,n} \\ x_{n-1,n} \\ x_{n,n-1} \end{pmatrix} = \begin{pmatrix} a & 0 & -c \\ 1 & 0 & 0 \\ 0 & 0 & 0 \end{pmatrix} \begin{pmatrix} x_{n-1,n} \\ x_{n-2,n} \\ x_{n-1,n-1} \end{pmatrix} + \begin{pmatrix} b & 0 & 0 \\ 0 & 0 & 0 \\ 1 & 0 & 0 \end{pmatrix} \begin{pmatrix} x_{n,n-1} \\ x_{n-1,n-1} \\ x_{n,n-2} \end{pmatrix} + \begin{pmatrix} \beta & 0 & 0 \\ 0 & 0 & 0 \\ 0 & 0 & 0 \end{pmatrix} \begin{pmatrix} \xi_{n,n} \\ \varphi_{n,n} \\ \phi_{n-1,n-1} \end{pmatrix} =$$
$$= A_1 \bar{x}_{n-e_1} + A_2 \bar{x}_{n-e_2} + B\bar{\xi}_{ij}.$$

В этом случае

$$E - \sum_{j=1}^{2} z^{-j} A_j = (E - z_1 A_1 - z_2 A_2) = \begin{pmatrix} 1 - ac z_1 - b z_2 & 0 & c z_1 \\ -z_1 & 1 & 0 \\ -z_2 & 0 & 1 \end{pmatrix}$$ и

подынтегральное выражение в (4.9) легко вычисляется:

$$V_x(m,n) = \frac{\beta^2}{(2\pi)^2} \iint_{QQ} \frac{x^{-m} y^{-n}}{(1 - ax - by + cxy)(xy - ay - bx + c)} \begin{pmatrix} 1 & 1/x & 1/y \\ x & 1 & x/y \\ y & y/x & 1 \end{pmatrix} dxdy$$

.

Здесь для простоты записей z_1 и z_2 заменены на x и y.

Наибольший интерес представляет элемент $V(n_1,n) = M[x_{m,n} x_{0,0}] = V(-n_1,-n)$ матрицы $V_x(m,n)$, т. е. КФ модели (4.10). Остальные элементы могут быть получены простым сдвигом. Учитывая симметричность, находим

$$V_x(m,n) = \frac{\beta^2}{(2\pi)^2} \iint_{QQ} \frac{1}{(1 - ax - by + cxy)(xy - ay - bx + c)} (x^m y^n)^{\pm} dxdy$$

. (4.11)

Будем вычислять этот интеграл с помощью вычетов, учитывая, что только второй множитель знаменателя имеет корни внутри единичного поликруга $K_2 = \{(x,y) : |x| \le 1, |y| \le 1\}$. Разлагая дробь в (4.11) на простейшие дроби по переменным x и y двумя способами, получаем

$$V_x(m,n) = \frac{\beta^2}{\gamma^2 (2\pi)^2} \iint_{QQ} \left[\frac{1}{x - x_1} - \frac{1}{x - x_2} \right] \left[\frac{1}{y - \frac{bx - c}{x - a}} - \frac{1}{y - \frac{ax - 1}{cx - b}} \right] (x^m y^n)^{\pm} dxdy =$$
$$= \frac{\beta^2}{\gamma^2 (2\pi)^2} \iint_{QQ} \left[\frac{1}{y - y_1} - \frac{1}{y - y_2} \right] \left[\frac{1}{x - \frac{cy - c}{y - b}} - \frac{1}{x - \frac{by - 1}{cy - a}} \right] (x^m y^n)^{\pm} dxdy,$$

(4.12)

где

$$x_1 = \frac{1}{2(a - bc)}(\varphi - \gamma), \quad x_2 = \frac{1}{x_1} = \frac{1}{2(a - bc)}(\varphi + \gamma), \quad y = \frac{1}{2(b - ac)}(\phi - \gamma), \quad y_1 = \frac{1}{y_1}(\phi + \gamma),$$
$$\varphi = 1 - c^2 + a^2 - b^2; \quad \phi = 1 - c^2 - a^2 + b^2; \quad \gamma = \sqrt{[(1+c)^2 - (a+b)^2][(1-c)^2 - (a-b)^2]}.$$

При этом $|x_1| < 1, \quad |y_1| < 1$. Кроме того, $\left| \frac{bx - c}{x - a} \right| < 1, \quad \left| \frac{ax - 1}{cx - b} \right| < 1$ при $|x| = 1$ и $\left| \frac{cy - c}{y - b} \right| < 1, \quad \left| \frac{by - 1}{cy - a} \right| > 1$ при $|y| = 1$. Следовательно, только первые дроби в скобках (4.12) дают ненулевые вычеты при последовательном вычислении интегралов. Опуская несложные выкладки, приведем итоговое выражение для КФ случайного поля:

$$V(m,n) = \sigma_x^2 x_1^{|m|} y_1^{|n|}, \quad mn \le 0,$$ (4.13)

где $\sigma_x^2 = V(0,0) = \beta^2 / \gamma$ – дисперсия поля. Таким образом, для получения поля с заданной дисперсией σ_x^2 следует взять $\beta = \sigma_x \sqrt{\gamma}$. Если $c = ab$, то $\gamma = (1 - a^2)(1 - b^2)$, $\beta = \sigma_x \sqrt{(1 - a^2)(1 - b^2)}$, $x_1 = a$, $y_1 = b$, что полностью совпадает с результатами, полученными для модели (3.4).

Если m и n имеют одинаковые знаки, т. е. $mn > 0$, то интеграл (4.11) дает более сложные выражения. Например, при неотрицательных m и n

$$V(1,n) = x_2 y_1^n - b(x_2 - x_1),$$
$$V(m,1) = x_1^m y_2 - a^m(y_2 - y_1),$$
$$V(2,n) = x_2^2 y_1^n + nb^n(c - ab)(x_2 - x_1) - b^n(x_2^2 - x_1^2),$$
$$V(m,1) = x_1^m y_2^2 + ma^m(c - ab)(y_2 - y_1) - a^m(y_2^2 - y_1^2).$$

В этом и аналогичных случаях удобнее вычислять значения $V(m,n)$ рекуррентно, исходя из уравнения КФ поля. Умножая общее линейное уравнение авторегрессии (3.3) на $x_{\bar{0}} - x_{0,0,\ldots,0}$ и находя математические ожидания, получим уравнение

$$V_x(\bar{i}) = \sum_{j=1}^{m} a_j V_x(\bar{i} + \bar{j}) + \beta M[x_{\bar{i}} \xi_i] \quad . \qquad (4.14)$$

Математическое ожидание $u_{\bar{i}} = M[x_{\bar{0}} \xi_{\bar{i}}]$ равно коэффициенту при $\xi_{\bar{i}}$ в разложении $x_{\bar{0}}$ по $\{\xi_{\bar{i}}\}$, т. е. в представлении (3.3) в виде взвешенных сумм. В частности, $u_{\bar{0}} = \beta, u_{\bar{i}} = 0$, если $x_{\bar{0}}$ предшествует $x_{\bar{i}}$ для данной развертки, так как в этом случае $x_{\bar{0}}$ и $\xi_{\bar{i}}$ независимы.

Связь соседних значений, схематически представленная на рис. 4.1,а, приводит к тому, что возмущения $\xi_{m,n}$ оказывают влияние на элементы поля в направлениях, изображенных на рис.4.1,б. Именно

поэтому $u_{m,n} \neq 0$ лишь при одновременном выполнении неравенств $m \leq 0$ и $n \leq 0$. Учитывая это замечание и вид (4.13) КФ, получаем:

$$V(m,n) = aV(m-1,n) + bV(m,n-1) - cV(m-1,n-1), \quad m,n > 0,$$

что вместе с граничными условиями

$$V(m,0) = \sigma_x^2 x_1^{|m|}, \quad V(0,n) = \sigma_x^2 y_1^{|n|}$$

позволяет последовательно вычислить необходимые значения КФ.

Как уже отмечалось, в случае $c = ab$ **изокорреляционными линиями** поля являются ромбы (рис.4.2,а,б). Если $c \neq ab$, то в области $mn \leq 0$ эти линии по-прежнему прямые, как это следует из (4.13). В области $mn > 0$ линии выпуклы при $c < ab$ (рис.4.2,в) и вогнуты при $c > ab$ (рис.4.2,г).

Таким образом, даже незначительное обобщение модели (3.4) (отказ от частного вида $c = ab$) позволяет получать СП с более широким классом КФ.

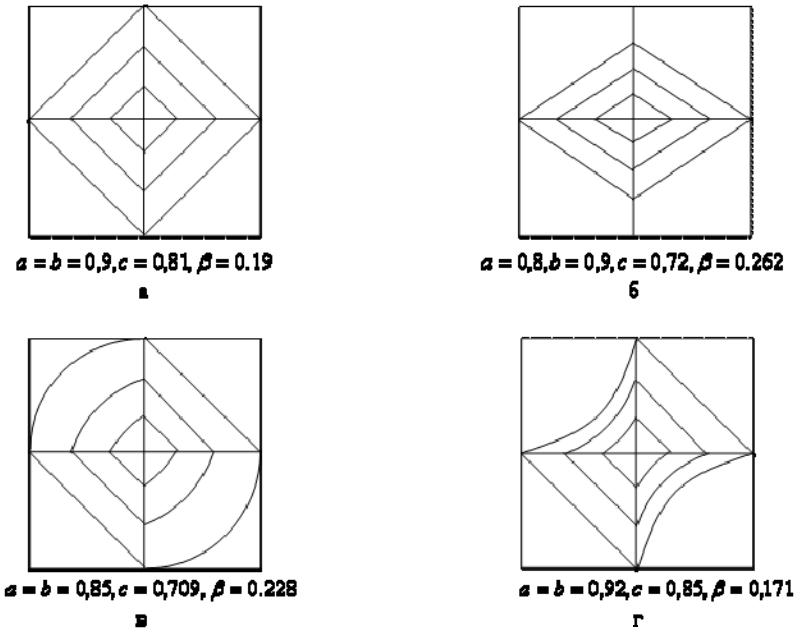
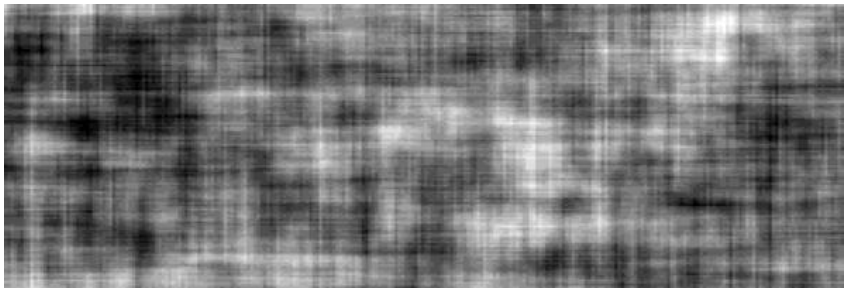
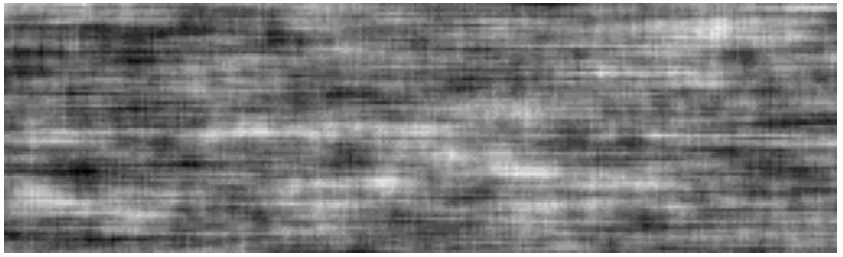


Рис.4.2.

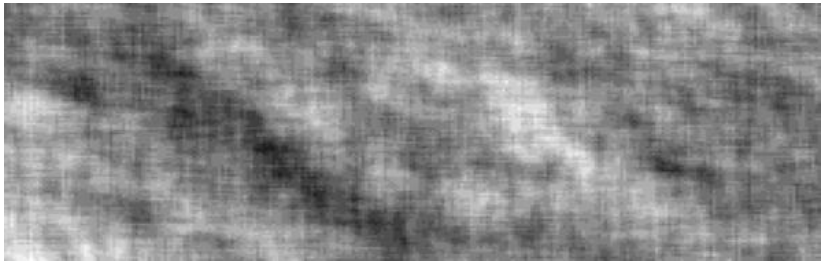
На рис.4.3,а –4.3,г приведены примеры имитированных И, порождаемых моделью (4.10) с $\sigma_{\varepsilon}^2 = 1$ и остальными параметрами, соответствующими рис.4.2,а –4.2,г. Реализация на рис.4.3,а напоминает клетчатую ткань. Эта особенность реализаций является проявлением анизотропии КФ. Действительно, КФ медленнее убывает вдоль координатных осей, чем по диагональ ным направлениям, поэтому И сильнее коррелировано вдоль осей и для реализаций характерно наличие продольных и поперечных полос .



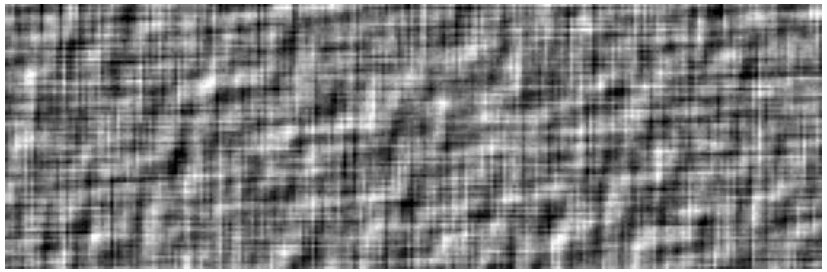
а



б



в



г

Рис.4.3.

Для И на рис.4.3,б характерны протяженные горизонтальные полосы, что объясняется большей корреляцией в горизонтальном направлении, чем в вертикальном (рис.4.2,б). В случае, когда сечение КФ имеет вид рис.4.2,в, сечения КФ уже довольно округлы, поэтому и на И (рис.4.3,в) анизотропия выражена слабее. С уменьшением коэффициента c сечения КФ становятся более вытянутыми из левого верхнего в правый нижний угол, поэтому на реализациях проявляются области, протяженные в этом направлении. Для рис.4.3,г характерна значительная коррелированность по направлению из левого нижнего в правый верхний угол, что объясняется видом сечений КФ на рис.4.2,г.

Четырехточечная модель

Рассмотрим теперь **четырехточечную** авторегрессионую модель

$$x_{m,n} = ax_{m-1,n} + bx_{m,n-1} - cx_{m-1,n-1} - dx_{m-1,n+1} + \beta\xi_{m,n} \qquad (4.15)$$

с шаблоном, представленным на рис.4.4,а.

Рис.4.4.

Аналогично предыдущему примеру, КФ порождаемого поля определяется выражением

$$V(m,n) = \frac{\beta^2}{(2\pi)^2}\iint_{qq}\frac{y(x^m y^n)^{\pm}}{(1-axy-by^2+cxy+dx)(xy-ay-bx+c+dy^2)}\,dxdy .$$

При $m = 0$ этот интеграл с помощью несложных преобразований дает следующий результат:

$$V(0,n) = \varphi y_1^{|n|} + \phi y_2^{|n|}, \qquad (4.16)$$

где

$$\varphi = \frac{\beta^2 y_1^2 y_2}{cd(1-y_1y_2)(1-y_1^2)(y_2-y_1)}; \quad \phi = \frac{\beta^2 y_1 y_2^2}{cd(1-y_1y_2)(1-y_2^2)(y_1-y_2)}$$

; y_1 и y_2 – корни уравнения

$$Fy^2 + 2Gy + H = 0 \qquad (4.17)$$

или величины, обратные этим корням, так, чтобы выполнялись неравенства $|y_1| < 1$ и $|y_2| < 1$. Коэффициенты уравнения (4.17)

определяются по формулам $u = -cd$, $v = a(c+d) - b$, $w = 1 - a^2 + b^2 - c^2 - d^2$, $p = \sqrt{w + 2u + 2v}$, $q = \sqrt{w + 2u - 2v}$, $g = p + q$, $h = \sqrt{g^2 - 16u}$, $F = g + h$, $G = p - q$, $H = g - h$. Выбирая коэффициенты β так, чтобы $\varphi + \phi = \sigma_x^2$, получаем поле с заданной дисперсией σ_x^2.

Таким образом, КФ вдоль горизонтальной координатной оси равна сумме двух экспонент. Анализируя направления, по которым оказывают влияние возмущения $\xi_{m,n}$ (рис.4.4,б), можно сделать вывод, что КФ удовлетворяет уравнению

$$V(m,n) = aV(m-1,n) + bV(m,n-1) - cV(m-1,n-1) - dV(m-1,n+1) \quad (4.18)$$

при положительных m и любых n, а также для $m \le 0$ при $n \ge 1 - m$. Решая это уравнение с граничными условиями (4.16), можно получить явное выражение для КФ:

$$V(m,n) = \varphi x_1^{|m|} y_1^{|n|} + \phi x_2^{|m|} y_2^{|n|}, \quad mn \le 0, \ m \le n \quad (4.19)$$

где $x_1 = (cy_1^2 - ay_1 + d) / y_1(by_1 - 1)$, $x_2 = (cy_2^2 - ay_2 + d) / y_2(by_2 - 1)$. Найденный результат справедлив в области M, ограниченной прямыми $m = 0$ и $m = -n$ (рис.4.4,в). Остальные значения КФ можно получить рекуррентно с помощью соотношений (4.18) и (4.19).

В качестве примера на рис. 4.5 приведены сечения КФ, соответствующей значениям $a = 0{,}2$; $b = 0{,}65$; $c = 0{,}2$; $d = -0{,}27$; $\beta = 0{,}13$.

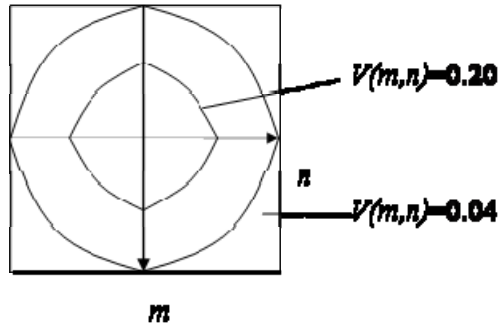


Рис.4.5.

Варьируя параметры модели (4.5), можно получать случайные поля с весьма широким классом КФ.

Как уже отмечалось, задача синтеза модели, порождающей случайное поле с заданной КФ, значительно сложнее задачи анализа модели. В рамках линейной авторегрессии задача синтеза сводится к определению подходящих коэффициентов уравнения (3.3). Например, выбирая пять коэффициентов уравнения (4.15), получим модель, порождающую поле с пятью заданными значениями КФ. Увеличивая размеры области локальных состояний, можно получить поля с более точным соответствием реальных изображений и рассмотренных моделей.

6.5. Тензорные модели случайных полей

Рассмотрим случайное поле $X = \{x_j^i : i \in I, \overline{j} \in \Omega\}$, заданное на (n+1)-мерной сетке $\Omega \times I$, где

$\Omega = \{\bar{j} = (j_1, j_2, j_3, \ldots, j_n)\}$ – n-мерная $M_1 \times M_2 \times \ldots \times M_n$-сетка; $I = \{i : i = 1,2,3\ldots\}$. Индекс i может интерпретироваться как время, поэтому сечение $x^i = \{x^i_{\bar{j}} : \bar{j} \in \Omega\}$ поля X будем называть i-м кадром. Покажем, что если поле X рассматривается как последовательность кадров $x_1, x_2, \ldots$, каждый из которых задан на n-мерной сетке W, то можно обобщить методы описания случайных последовательностей на случайные поля.

Пусть последовательность кадров описывается стохастическим разностным уравнением

$$x^i = \varphi^i(x^{i-1}) + \mathcal{G}^i(x^{i-1})\xi^i, \quad i = 1,2,\ldots, \tag{5.1}$$

где $\{\xi^i_{\bar{j}} : i \in I, \bar{j} \in \Omega\}$ – порождающее стандартное гауссовское поле; $\xi^i = (\xi^i_{\bar{j}} : \bar{j} \in \Omega)$ – i-й кадр этого поля;

$\varphi^i(x^{i-1}) = \{\varphi^i_{\bar{j}}(x^{i-1}) : \bar{j} \in \Omega\}$ _ $M_1 \times M_2 \times \ldots \times M_n$ _

$\mathcal{G}^i(x^{i-1}) = \{\mathcal{G}^i_{\bar{j}\bar{l}}(x^{i-1}) : \bar{j}, \bar{l} \in \Omega\}$ матричная функция; – тензоры ранга 2n с двумя групповыми индексами, формирующие возмущающую компоненту i-го кадра из ξ^i по правилу умножения

$\mathcal{G}^i\xi^i = \{\mathcal{G}^i_{\bar{j}}\}\{\xi^i_{\bar{l}}\} = \{\sum \mathcal{G}^i_{\bar{j}\bar{l}}\xi^i_{\bar{l}}\}$ тензоров . Транспонирование этого кадра заключается в перестановке его групповых индексов:

$\mathcal{G}^{iT}_{\bar{j}} = \mathcal{G}^i_{\bar{l}}$. Отметим, что верхний индекс i означает номер кадра,

т. е. в нашей интерпретации – время, поэтому i не считается немым индексом и суммирование по нему не производится.

Модель (5.1) позволяет описывать весьма широкий класс марковских последовательностей случайных кадров. В частности, линейная модель

$$x^i = P^i x^{i-1} + \mathcal{G}^i\xi^i, \tag{5.2}$$

где тензоры P^i и $\mathcal{G}^i$ не зависят от x^{i-1}, описывает гауссовскую последовательность.

Для этого поля КФ есть **ковариационная пространственная матрица,** определяемая естественным образом:

$$V_x(i,j) = M[(x^i - m^i) \times (x^j - m^j)],$$
$$V_x(i) = V_x(i,i), \tag{5.3}$$

где $m^i = M[x^i] = \{M[x^i_{\bar{j}}] : \bar{j} \in \Omega\}$, а символ «$\times$» обозначает внешнее произведение матриц. Таким образом, $V_x(i)$ и $V^i_\xi = M[\xi^i \times \xi^i]$ являются симметричными $M_1 \times M_2 \times \ldots \times M_n \times M_1 \times M_2 \times \ldots \times M_n$-матрицами. Для полного определения случайного поля с помощью уравнения состояния (5.1) необходимо задать закон распределения начального кадра x_0. Часто это распределение является гауссовским со средним m^0 и ковариационной матрицей V^0_x. Совместную ПРВ первых кадров представим следующим образом:

$$w(x^0, x^1, \ldots, x^k) = w(x^0) \prod_{i=1}^{k} w(x^i \mid x^{i-1}, x^{i-2}, \ldots, x^1)$$

. (5.4)

Из (5.1) следует, что в гауссовском случае

$$w(x^i \mid x^{i-1}, x^{i-2}, \ldots, x^1) = w(x^i \mid x^{i-1}) = [(2\pi)^M \det V_x^i]^{-\frac{1}{2}} \exp\left(-\tfrac{1}{2}\left\|x^i - \varphi(x^{i-1})\right\|^2_{(V_x^i)^{-1}}\right),$$

(5.5)

где $V_x^i = g^i(x^{i-1}) V_\xi^i g^{iT}(x^{i-1})$; $\|a\|_V^2 = aVa$;
$M = M_1 \times M_2 \times \ldots \times M_n$.

Подставляя (5.5) в (5.4), получим следующее выражение для совместной ПРВ:

$$w(x^0, \ldots, x^k) = [\prod_{i=0}^{k} (2\pi)^M \det V_x^i]^{-\frac{1}{2}} \exp\left(-\tfrac{1}{2}\sum_{i=0}^{k}\left|x^i - \varphi^i(x^{i-1})\right|^2_{(V_\xi^i)^{-1}}\right),$$

(5.6)

где $j^1(x^0) = m^0$. Из совместных ПРВ (5.6), вообще говоря, можно найти среднее значение m_i и КФ

$V_x(i,j) = M[(x^i - m^i) \times (x^j - m^j)]$, но выполнить практически необходимые для этого вычисления сложно. Удобнее воспользоваться известными приближенными рекуррентными соотношениями для векторных марковских последовательностей, обобщив их на рассматриваемые **тензорные модели**.

Для этого используем разложение функции $\varphi^i(x^{i-1})$ в тензорный ряд Тейлора, предполагая достаточную гладкость и ограничиваясь линейными членами:

$$\varphi^i(x^{i-1}) \approx \varphi^i(m^{i-1}) + \frac{\partial \varphi^i(m^{i-1})}{\partial m^{i-1}}(x^{i-1} - m^{i-1})$$

. (5.7)

Подставляя (5.7) в (5.1) и усредняя, находим приближенное рекуррентное соотношение

$$m^i \approx \varphi^i(m^{i-1}) \qquad (5.8)$$

для средних значений. Подставляя (5.7) и (5.8) в (5.3) и учитывая независимость xi и xi, получаем

$$V_x(i) = \frac{\partial \varphi^i(m^{i-1})}{\partial m^{i-1}} V_x(i-1) \frac{\partial \varphi^{iT}(m^{i-1})}{\partial m^{i-1}} + M[g^i(x^{i-1}) V_\xi^i g^{iT}(x^{i-1})]$$

. (5.9)

Разложим $g^i V_\xi^i g^{iT}$ в тензорный ряд Тейлора с точностью до линейных членов:

$$g^i(x^{i-1}) V_\xi^i g^{iT}(x^{i-1}) = g^i(m^{i-1}) V_\xi^i g^{iT}(m^{i-1}) + 2 g^i(m^{i-1}) V_\xi^i \frac{\partial g^{iT}(m^{i-1})}{\partial (m^{i-1})}(x^{i-1} - m^{i-1}).$$

Подставляя это разложение в (5.9), получим приближенное рекуррентное соотношение для КФ:

$$V_x(i) = \frac{\partial \varphi^T(m^{i-1})}{\partial (m^{i-1})} V_x(i-1) \frac{\partial \varphi^{iT}(m^{i-1})}{\partial (m^{i-1})} + g^i(x^{i-1}) V_\xi^i g^{iT}(x^{i-1}).$$

(5.10)

В случае линейной модели (5.2) рекуррентные соотношения (5.8) и (5.10) будут точными:

mi = 0, $$V_x(l) = P^l V_x(l-1) P^{lT} + \vartheta^l V_\xi^l \vartheta^{lT}$$. (5.11)

Особый интерес представляет модель (5.2) с постоянными тензорами $P^l = P$ и $\vartheta^l = \vartheta$, для которой

$$V_x(l, l+k) = P^{(k)} V_x(l,l) = P^{(k)} V_x(l)$$, (5.12)

где (k) означает возведение в k–ю степень. Если корни характеристического уравнения $\det(\lambda E - P) = 0$ по модулю меньше единицы, то P(k) ® 0 при k ® ¥, и из (5.12) находим, что $V_x(l, l+k) \to 0$ при k ® ¥.

Записывая (5.2) в виде $(E - zP)x^l = \vartheta \xi^l$ и производя выкладки, аналогичные (4.2)–(4.9), получаем **тензорный спектр** стационарного поля

$$V_x(z) = [E - zP]^{-1} \vartheta V_\xi \vartheta^T [E - z^{-1} P^T]^{-1}$$

и выражение для КФ

$$V_x(0,k) = \frac{1}{2\pi i} \int_{C_1} [E - zP]^{-1} \vartheta V_\xi \vartheta^T [E - z^{-1} P^T]^{-1} z^{k-1} dz$$. (5.13)

Из (5.13) достаточно найти $V_x = V_x(0,0)$, остальные значения получаются из уравнения (5.12), которое в стационарном случае принимает форму $V_x(l, l+k) = P^{(k)} V_x$. Для нахождения V_x можно вместо интегральной формулы (1.50) использовать предел (1.48) при

i ® ¥.

$$V_x = PVP^T + \vartheta V_\xi \vartheta^T$$. (5.14)

Уравнение (5.14) представляет собой неособенную систему линейных уравнений относительно компонент тензора V_x.

Аналогичным образом могут быть обобщены авторегрессионые скалярные и векторные модели, порождающие марковские последовательности более высоких порядков. Это дает возможность описания последовательностей многомерных кадров, в которых каждый кадр зависит от m непосредственно предшествующих ему кадров.

6.6. Волновые модели случайных полей

В двух предыдущих подразделах приведено решение задачи анализа для авторегрессионых и тензорных моделей СП. Принципиальные решения этой задачи имеются и для других моделей полей, но воспользоваться ими можно только в относительно простых частных случаях, так как конкретизация решения связана с громоздкими выкладками и вычислениями, например, с действиями над матрицами и тензорами больших размеров и вычислением многомерных интегралов по комплексным переменным.

Еще более сложной в конструктивном отношении является задача синтеза. Вместе с тем ее решение необходимо, например, для имитации СП с заданной КФ.

Рассмотрим **волновую модель** СП, являющуюся обобщением ряда других моделей и позволяющую эффективно решать задачи корреляционного анализа и синтеза. Эта модель достаточно проста и может служить основой для имитации СП с заданной КФ без увеличения числа параметров модели.

В волновой модели СП определяется равенством

$$x^t_{\bar{j}} = \sum_{\{k:\tau_k \le t\}} f((\bar{j},t),(\bar{u}_k,\tau_k),\bar{\omega}_k) \quad , \qquad (6.1)$$

где (n+1)-мерная область определения $\{(\bar{j},t)\}$ поля может быть сеточной или непрерывной; $\{(\bar{u}_k,\tau_k)\}$ – дискретное **поле случайных точек** (ПСТ) в (n+1)-мерном непрерывном пространстве; t и tk интерпретируются как время; $\bar{\omega}_k$ – случайный вектор параметров функции f.

Это поле можно представить как результат воздействия случайных возмущений или волн $f((\bar{j},t),(\bar{u}_k,\tau_k),\bar{\omega}_k)$, возникающих в случайных местах $\bar{u}_k$ в случайные моменты времени τ_k и изменяющихся по заданному закону во времени и пространстве.

Выбор функции f, параметров ПСТ и $\bar{\omega}$ позволяет получить широкий класс полей, включающий в себя следующие модели.

1. Пуассоновские поля: при

$f((\bar{j},t),(\bar{u}_k,\tau_k),\bar{\omega}_k) = \delta((\bar{j},t)-(\bar{u}_k,\tau_k))$, где d – символ Кронекера и $(\bar{u}_k,\tau_k)$ – пуассоновское ПСТ.

2. Многомерный фильтрованный пуассоновский процесс: при,

$f((\bar{j},t),(\bar{u}_k,\tau_k),\bar{\omega}_k) = g((\bar{j}-\bar{u}_k t-\tau_k),\xi_k)$ где $\{\xi_k\}$ – система скалярных СВ. Эта модель порождает только стационарные

однородные поля, а образующие волны могут отличаться друг от друга только одним параметром ξ_k.

3. Модель взвешенных сумм: при

$f((\bar{j},t),(\bar{u}_k,\tau_k),\bar{\omega}_k) = g((\bar{j},t),(\bar{u}_k,\tau_k))\xi_k$, где $\{(\bar{u}_k,\tau_k)\}$ – совокупность всех узлов сетки и g – соответствующие веса случайных величин ξ_k.

4. Модель случайных блужданий: ПСТ описывает случайное блуждание (возможно, с возникновением и исчезновением) совокупности волн, а выбор $\bar{\omega}_k$ определяет динамику формы и интенсивности волн. Такие модели можно применить, например, для **имитации изображения движущихся облаков**.

Рассмотрим частный случай волновой модели, для которой корреляционные задачи анализа и синтеза легко решаются. Пусть

$$f((\bar{j},t),(\bar{u}_k,\tau_k),\bar{\omega}_k) = g(\rho_k / R_k)\exp(-\mu|t-\tau_k|)\xi_k \quad , \qquad (6.2)$$

где ПСТ – пуассоновское с постоянной плотностью l; $\rho_k = |\bar{j}-\bar{u}_k|$ – расстояние между $\bar{j}$ и $\bar{u}_k$; $\{\rho_k\}$ – система независимых неотрицательных одинаково распределенных СВ с ПРВ w(a); $\{\xi_k\}$ – система независимых одинаково распределенных СВ. В этом случае волны неподвижны, независимы между собой, имеют сферические сечения по пространству и экспоненциально затухают со временем; система $\{\xi_k\}$ определяет интенсивность волн, а $\{\rho_k\}$ – их пространственный масштаб.

Порождаемое поле X, очевидно, стационарно, однородно, имеет нулевое среднее и изотропную по пространству КФ

$$V(\rho,t) = M[x^{t}_{(-\rho/2,0,0,\ldots,0)} x_{(\rho/2,0,0,\ldots,0)}] \quad (6.3)$$

Учитывая, что в данном случае слагаемые в (6.1) не коррелированы и элементарное событие $\Delta A = \Delta V = \Delta_1 \Delta_2 \ldots \Delta_d \Delta\tau$ {в элементе возникла точка ПСТ, которой соответствует волна с пространственным масштабом a из элемента Da} имеет вероятность $P(\Delta A) = \lambda \Delta V \omega(\alpha) \Delta\alpha$, выразим (6.3) через интеграл по переменным t, α, $j_1,\ldots,j_n$. После интегрирования по t получаем

$$V(\rho,t) = \frac{\lambda}{2\mu} e^{-\mu t} \left[\int_{-\infty}^{\infty} \ldots \int_{-\infty}^{\infty} g\left(\frac{\sqrt{(U_1 - \rho/2)^2 - j_2^2 + \ldots + j_n^2}}{\alpha} \right) \times \right.$$

$$\left. \times g\left(\frac{\sqrt{(U_1 + \rho/2)^2 + j_2^2 + \ldots + j_n^2}}{\alpha} \right) \omega(\alpha) dj_1 \ldots dj_n d\alpha \right.$$

Этот (n+1)-кратный интеграл сводится к однократному

$$V(\rho,t) = \frac{c^2 \pi^{n/2} \lambda}{2^{n+1} \mu} e^{-\mu t} \int_0^{\infty} \alpha^n \exp\left(-\frac{\rho^2}{\alpha^2} \right) \omega(\alpha) d\alpha \quad (6.4)$$

если выбрать g(y) = c exp(-2y2).

При r = t = 0 из (6.4) находим дисперсию поля

$$\sigma_1^2 = \frac{c^2 \pi^{n/2} \lambda}{2^{n+1} \mu} M[R^n] \quad (6.5)$$

пропорциональную плотности l ПСТ, эффективному интервалу 1/m затухания волн и среднему значению n-й степени пространственного масштаба R.

Имитация дискретного поля на n-мерной сетке $\{\bar{j}\}$ с шагом квантования D_t по времени на основе модели рассмотренного частного вида может быть осуществлена с помощью следующего алгоритма. В начальный момент $t_0 = 0$ значения поля во всех узлах равны нулю. В каждый последующий момент $t_m = mD_t$ на непрерывном пространстве или на сетке, несколько перекрывающей $\{\bar{j}\}$, формируется пуассоновское ПСТ с плотностью lD_t. В каждой сформированной точке ПСТ $\bar{\alpha}_k$ разыгрываются СВ x_k и R_k, после чего производится преобразование

$$x^{t_m}_{\bar{j}} = x^{t_{m-1}}_{\bar{j}} \exp(-\mu \cdot \Delta t) + \sum_k g(\rho_k / R_k) \xi_k \quad (6.6)$$

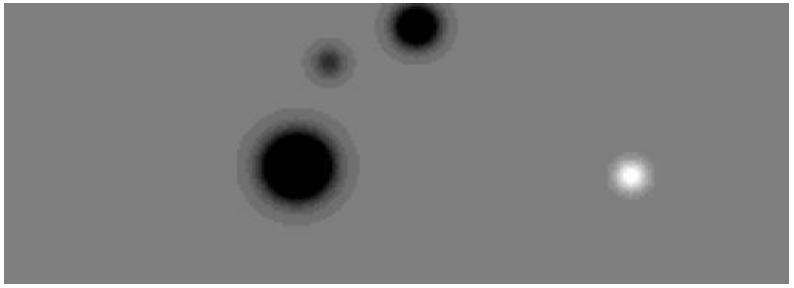
всех значений поля на сетке $\{\bar{j}\}$. При таком моделировании в (6.6) можно учитывать только достаточно большие по сравнению с уровнем квантования слагаемые. Достоинством такого алгоритма является его рекуррентность, что позволяет легко реализовать имитацию поля на ЭВМ.

На рис. 6.1 приведены реализации полей, полученных с помощью описанного алгоритма при различных входящих в модель параметрах. На рис.6. 1,а показан первый кадр поля, на котором отчетливо видны четыре волны с квадратично экспоненциальным сечением.

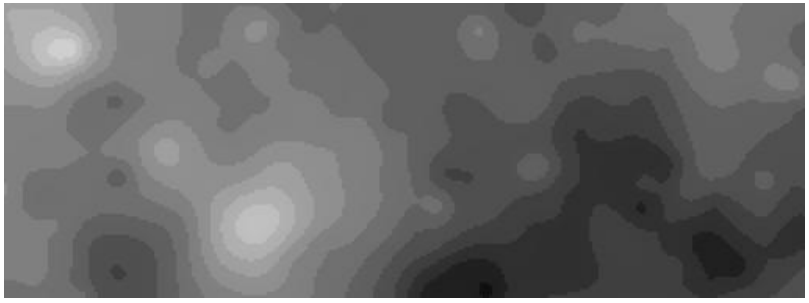
С течением времени волн становится все больше, они налагаются друг на друга, создавая плавное изображение. На рис.6.1,б показан двадцатый кадр этого процесса. По прошествии временного интервала порядка mD_t=1/m характер И практически не меняется – поле устанавливается. Происходит это потому, что волны затухают при многократном умножении на exp(mD_t) в (6.6). Основной вклад в формирование И вносят волны, возникающие на последних $m = 1/mD_t$ кадрах.

Рис.6.1,в получен при вырожденном распределении (R = 5), поэтому изображение выглядит проще по сравнению с предыдущим, так как состоит из волн, отличающихся друг от друга только интенсивностью. Поэтому на нем имеются образования примерно одинаковых размеров, а на рис.6.1,б присутствуют разномасштабные образования.

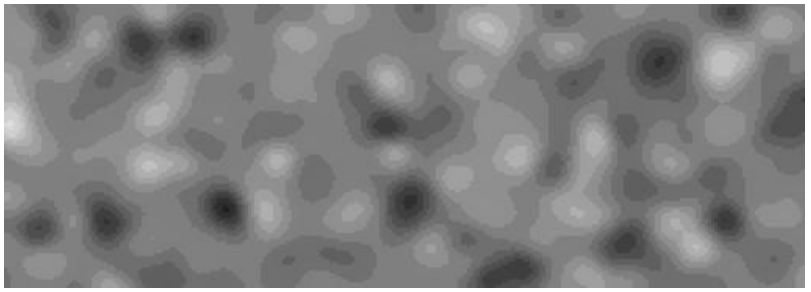
Если в формуле (6.1) функция g(y) кусочно-постоянна, то и реализации поля будут кусочно-постоянными. Пример реализации такого поля для функции g(y) = 1 при $|y| \le 1$ и g(y) = 0 при $|y| > 1$ приведен на рис.6.1,г, И является результатом наложения кругов различных диаметров и интенсивностей. На границе кругов наблюдаются контрастные переходы.



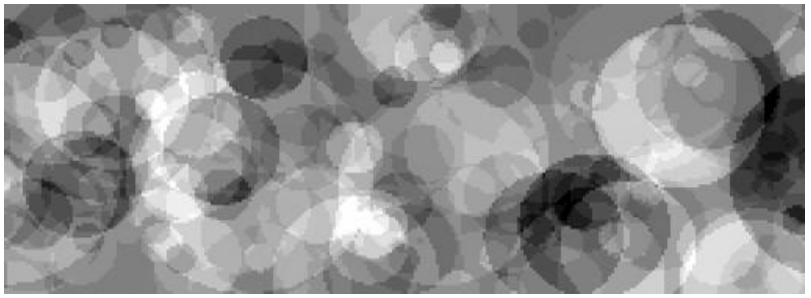
а



б



в



г

Рис.6.1.

Каждое значение x_i поля является суммой случайного числа СВ, поэтому поле, вообще говоря, не будет гауссовским даже при гауссовских $\{\xi_k\}$. Однако с ростом параметра модели $lM[R_n]/m$ в сумме (6.1) возрастает количество слагаемых с близкими распределениями, и поле нормализуется.

Рассмотрим теперь решение корреляционных задач анализа и синтеза. Из (6.4) следует, что построенное поле имеет экспоненциальную НКФ $e^{-\mu t}$ по времени и НКФ

$$r(\rho) = \frac{1}{M[R^n]} \int_0^\infty \alpha^n \exp(-\rho^2/\alpha^2)\omega(\alpha)d\alpha \qquad (6.7)$$

по пространству. Таким образом, при решении задачи анализа, когда ПРВ $\omega(\alpha)$ задана, искомая НКФ может быть найдена аналитически или численным интегрированием. При решении задачи синтеза, когда НКФ r(r) задана, необходимо решить интегральное уравнение (6.7) относительно неизвестной ПРВ $\omega(\alpha)$.

Выполняя в (6.7) замену x = a-2, получим выражение НКФ

$$r(\rho) = \frac{1}{2M[R^n]} \int_0^\infty x^{-\frac{n+2}{2}} \omega(x^{-\frac{1}{2}}) \exp(-\rho^2 x)dx \qquad (6.8)$$

через преобразование Лапласа функции $f(x) = x^{-(n+2)/2}\omega(x^{-1/2})$. Из (6.8) следует, что если $f(x)$ и $F(\rho)$ – оригинал и изображение при преобразовании Лапласа, то с точностью до постоянного множителя $r(\rho) = F(\rho^2)$ и $\omega(\alpha) = \alpha^{n+2} f(\alpha^{-2})$. Это обстоятельство позволяет использовать теорию преобразования Лапласа для решения поставленных задач анализа и синтеза. Например, при равномерной ПРВ $\omega(\alpha)$ на отрезке [a,b] получаем

$$r(\rho) = (n+1)\rho^{n+1}[\Gamma(-\frac{n+1}{2}, \rho^2 b^{-2}) - \Gamma(-\frac{n+1}{2}, \rho^2 a^{-2})]/2(b^{n+1} - a^{n+1})$$

,

при $\omega(\alpha) = 2\alpha^{2k+n}\alpha^{-(2k+n+1)}\exp(-\alpha^2/\alpha^2)/\Gamma(k+n/2)$ находим $r(\rho) = [a^2/(a^2+\rho^2)]^k$,

а при $\omega(\alpha) = 2\alpha^n b^n \alpha^{-(n+1)}[\exp(-a^2/\alpha^2) - \exp(-b^2/\alpha^2)]/(b^2 - a^2)\Gamma(n/2)$ имеем $r(\rho) = [\ln((b^2+\rho^2)/(a^2+\rho^2))]/[\ln(b^2/a^2)]$.

Поскольку найти аналитическое решение задачи синтеза удается не всегда, рассмотрим метод ее приближенного решения. Из (6.7) следует, что при вырожденном распределении (R=a=const) получаем КФ $\exp(-\rho^2/\alpha^2)$. Пусть теперь задана произвольная невозрастающая НКФ $r(\rho)$. Аппроксимируем ее с достаточной точностью суммой гауссоид с положительными коэффициентами: $r(\rho) \approx h(\rho) = \sum_i q_i \exp(-\rho^2/\alpha_i^2)$, где $\sum_i q_i = 1$, так как $r(0) = 1$. Тогда при дискретном распределении $P(R = \alpha_i) = k^{-1} q_i/\alpha_i^n$, где $k = \sum q_i/\alpha_i^n$, порождаемое поле будет иметь НКФ, в точности равную $h(\rho)$. Таким образом, построенная модель позволяет приближенно решать задачу синтеза с помощью вариации только распределения вероятностей масштаба R.

Отметим, что волновая модель может порождать и анизотропные поля. Для этого волны должны иметь форму, отличную от сферической. Если, например, в (6.2) положить $\rho_k = \sqrt{(j - \vec{u}_k)^T A (j - \vec{u}_k)}$, где A – положительно определенная матрица, то волны будут иметь эллипсоидальную форму с заданной ориентацией. Рассмотренному

выше случаю сферических волн соответствует A = E. В общем случае в формулах (6.4) и (6.5) добавится дополнительный множитель (det A)-1/2, равный произведению полуосей эллипсоида $\vec{j}^T A\vec{j} = 1$. При этом все точки поля на эллипсоиде $\sqrt{\vec{j}^T A\vec{j}} = \rho$ будут иметь ковариацию (6.4) с его центром, т. е. поле будет иметь эллипсоидальную анизотропию.

6.7. Векторные случайные поля

Векторные СП применяются в различных приложениях, например, в следующих.

1) Информационное СП может быть векторным, например, поле скоростей, спектрозональные (цветные) И, поле межкадровых смещений И и т.д.

2) Скалярное СП можно описать как векторное для упрощения уравнений этого описания. Например, мы уже применяли такой подход для уменьшения числа слагаемых в правой части авторегрессионной модели (количество слагаемых доводится до размерности СП).

Будем рассматривать векторные СП $X = \{\bar{x}_\Gamma : \Gamma \in \Omega\}$, заданные на конечной или бесконечной целочисленной n-мерной сетке W. Значения СП есть $\bar{x}_\Gamma = (x_{\Gamma 1}, x_{\Gamma 2}, \ldots, x_{\Gamma m})^T$ – векторы размерности m. В целом СП $X = \{x_{\Gamma,k} : \Gamma \in \Omega, k = \overline{1,m}\}$ можно рассматривать и как совокупность скалярных СВ, которая может быть определена различными способами.

6.7.1. Авторегрессионные модели векторных случайных полей

Рассмотрим линейную авторегрессионную модель векторного СП вида

$$\bar{x}_\Gamma = \sum_{j \in D} A_j \bar{x}_{\Gamma+j} + B\bar{\xi}_\Gamma, \qquad (7.1)$$

где A_j и B – квадратные матрицы; $\xi = \{\xi_\Gamma\}$ – порождающее стандартное СП, состоящее из независимых векторов с независимыми компонентами; D – множество индексов. Если сетка W ограничена (хотя бы частично), то нужно еще задать начальные условия, позволяющие воспользоваться уравнением (7.1). При подходящих начальных условиях (или на неограниченной сетке) порождаемое поле однородно, т. е. его КФ $V_x(\vec{i},\vec{j}) = M[x_i x_j^T]$ зависит только от взаимного расположения узлов $\vec{i}$ и $\vec{j}$: $V_x(\vec{i},\vec{j}) = V_x(0,\vec{j}-\vec{i})$. Поэтому будем использовать обозначение

$$V_x(\vec{i}) = V_x(0,\vec{i}). \qquad (7.2)$$

Отметим, что

$$V_x(-\vec{i}) = V_x^T(\vec{i}). \qquad (7.3)$$

Можно показать, что КФ порожденного поля может быть найдена из выражения, аналогичного случаю скалярных СП:

$$V_x(\vec{i}) = \frac{1}{(2\pi i)^n}\int_{\Omega_n}\left[E - \sum_{j\in D} z^j A_j\right]^{-1} BB^T\left[E - \sum_{j\in D} z^{-j} A_j^T\right]^{-1} z^{-\vec{i}-\bar{1}}\,dz, \qquad (7.4)$$

где i – мнимая единица;

$\bar{z}^{\vec{j}} = (z_1,\ldots,z_n)^{(j_1,\ldots,j_n)} = z_1^{j_1} z_2^{j_2}\ldots z_n^{j_n}$; $\bar{1} = (1,\ldots,1)$;

$dz = dz_1 dz_2 \ldots dz_n$; $C_n = \{|z_1| = 1,\ldots, |z_n| = 1\}$ _

единичная полиокружность (прямое произведение n единичных окружностей); E – единичная матрица. Таким образом, правая часть (7.4) есть n-кратный интеграл по n комплексным переменным $z_1,\ldots,z_n$, причем интегрирование каждого элемента подынтегральной матрицы (являющейся функцией n комплексных переменных) проводится независимо от остальных элементов. Так что в (7.4) после вычисления подынтегральной матрицы нужно вычислить много интегралов, что представляет собой громоздкую операцию.

Авторегрессионные модели с матрицами вида H(a,b)

Анализ модели (7.1) значительно упрощается, если в ней используются матрицы вида H(a,b). Рассмотрим векторную авторегрессионную модель (7.1) частного вида

$$\bar{x}_t = \sum_{j\in D} H(a_j,b_j)\bar{x}_{t+j} + H(\alpha,\beta)\bar{\varepsilon}_t \,, \qquad (7.5)$$

т. е. $A_j=H(a_j,b_j)$ и $B=H(a,b)$. Вид матричных коэффициентов в (7.5) означает, что k-я компонента x_{jk} вектора $\bar{x}_j$ формируется следующим образом. Соответствующая k-я компонента $x_{j+a,k}$ имеет вес $a_j + b_j$, а все остальные компоненты вектора $\bar{x}_{j+a}$ имеют веса b_j. С аналогичными весами a + b и b входят компоненты возмущающего вектора $\bar{\varepsilon}_j$. Такая модель соответствует реалистичному предположению о том, что на k-ю компоненту вектора $\bar{x}_j$ как-то влияет k-я компонента вектора $\bar{x}_{j+a}$, а все остальные компоненты вектора $\bar{x}_{j+a}$ влияют на $\bar{x}_j$ одинаково. Аналогична и роль возмущений.

Вычислим подынтегральное выражение в (7.4) для нахождения КФ порождаемого поля, используя свойства матриц H(a,b) и их компонент H_1 и H_2 :

$$BB^T = H(\alpha,\beta)H^T(\alpha,\beta) = H(\alpha,\beta)H(\alpha,\beta) = H^2(\alpha,\beta) = (\alpha+m\beta)^2 H_1 + \alpha^2 H_2;$$

$$\left[E - \sum_{j\in D} z^j H(a_j,b_j)\right]^{-1} = \left[H(1,0) - \sum_{j\in D} z^j H(a_j,b_j)\right]^{-1} = \left[H\left(1 - \sum_{j\in D} z^j a_j, -\sum_{j\in D} z^j b_j\right)\right]^{-1} =$$

$$= \frac{1}{1-\sum(a_j - mb_j)z^j} H_1 + \frac{1}{1-\sum z_j z^j} H_2,$$

где $H_1 = \frac{1}{m} I, H_2 = E - \frac{1}{m} I$. Подставляя эти выражения в (7.4) и учитывая, что $H_1^2 = H_1$, $H_2^2 = H_2$, $H_1 H_2 = H_2 H_1 = 0$, получаем:

$$V_x(l) = \frac{1}{(2\pi i)^n} \int_{\Omega} \frac{(\alpha+m\beta)^2 z^{l-1}}{(1-\sum_{j\in D}(a_j+mb_j)z^j)(1-\sum_{j\in D}(a_j+mb_j)z^{-j})} dz H_1 +$$

$$+ \frac{1}{(2\pi i)^n} \int_{\Omega} \frac{(\alpha+m\beta)^2 z^{l-1}}{(1-\sum_{j\in D} a_j z^j)(1-\sum_{j\in D} a_j z^{-j})} dz H_2 \qquad (7.6)$$

где интегралы берутся только от скалярных (комплексных) функций. Эти интегралы после их вычисления являются скалярными величинами, поэтому (7.6) можно представить в виде

$$V_x(l) = V_1(l) H_1 + V_2(l) H_2, \qquad (7.7)$$

где

$$V_1(\bar{i}) = \frac{(\alpha+m\beta)^2}{(2\pi i)^n}\int_{\Gamma}\frac{z^{\bar{i}-\bar{1}}}{(1-\sum_{j\in D}(a_j+mb_j)z^j)(1-\sum_{j\in D}(a_j+mb_j)z^{-j})}dz$$

,

$$V_2(\bar{i}) = \frac{\alpha^2}{(2\pi i)^n}\int_{\Gamma}\frac{z^{\bar{i}-\bar{1}}}{(1-\sum_{j\in D}a_jz^j)(1-\sum_{j\in D}a_jz^{-j})}dz$$

. (7.8)

Из (7.7) следует, что КФ $V_x(\bar{i})$ есть матрица вида $H(a(\bar{i}),b(\bar{i}))$, в которой $a(\bar{i})$ и $b(\bar{i})$ удовлетворяют системе $\begin{cases} a(\bar{i})+mb(\bar{i}) = V_1(\bar{i}) \\ a(\bar{i}) = V_2(\bar{i}) \end{cases}$, откуда $a(\bar{i}) = V_2(\bar{i})$,

$b(\bar{i}) = \frac{1}{m}(V_1(\bar{i}) - V_2(\bar{i}))$, т. е.

$V_x(\bar{i}) = H(V_2(\bar{i}), \frac{1}{m}(V_1(\bar{i}) - V_2(\bar{i})))$ или непосредственно из (7.7)

$$V_x(\bar{i}) = \frac{1}{m}\begin{pmatrix} V_1+(m-1)V_2 & V_1-V_2 & \dots & V_1-V_2 \\ V_1-V_2 & V_1+(m-1)V_2 & \dots & V_1-V_2 \\ \dots & \dots & \dots & \dots \\ V_1-V_2 & V_1-V_2 & \dots & V_1+(m-1)V_2 \end{pmatrix}$$

. (7.9)

То, что $V_x(\bar{i})$ является матрицей вида H(a,b), следует и из вида модели (7.5).

Рассмотрим теперь представление $V_1(\bar{i})$ и $V_2(\bar{i})$ выражениями (7.8). Каждое из этих выражений является скалярным вариантом выражения (7.4). Функция $V_1(\bar{i})$ получается, если в (7.4) взять E=1, $A_j = a_j + mb_j, B = \alpha + m\beta$, а $V_2(\bar{i})$ получается при E=1, $A_j = a_j, B = \alpha$. К таким выражениям мы пришли бы, если бы в (7.1) взяли соответствующие скалярные значения A_j и B, т.е. если бы рассмотрели не векторные, а скалярные авторегрессионные модели СП

$$y_{1,\bar{i}} = \sum_{j\in D}(a_j+mb_j)y_{1,\bar{i}+j} + (\alpha+m\beta)\xi_{1,\bar{i}}$$

, (7.10)

$$y_{2,\bar{i}} = \sum_{j\in D}a_jy_{2,\bar{i}+j} + \alpha\xi_{1,\bar{i}}$$

, (7.11)

порождающие два скалярных СП $Y_1 = \{y_{1,\bar{i}} : \bar{i} \in \Omega\}$ и $Y_2 = \{y_{2,\bar{i}} : \bar{i} \in \Omega\}$. Эти СП имеют КФ $V_1(\bar{i})$ и $V_2(\bar{i})$ соответственно.

Пример. Пусть m=n=2, т.е. рассмотрим плоское СП с двумерными векторами $\bar{x}_{\bar{i}} = (x_{\bar{i},1}, x_{\bar{i},2})$. Это может быть, например, поле смещений между двумя плоскими И. Пусть порождающая модель имеет вид

$$\bar{x}_{\bar{i}} = H(a_1,b_1)\bar{x}_{i-1,j} + H(a_2,b_2)\bar{x}_{i,j-1} + H(\alpha,\beta)\bar{\xi}_{i,j}$$

, (7.12)

где $\bar{\xi}_{i,j} = (\xi_{i1}, \xi_{i2})^T$. В этом случае модели скалярных СП Y_1 и Y_2 в (7.10) и (7.11) принимают вид

$$y_{1i,j} = (a_1 + 2b_1)y_{1,i-1,j} + (a_2 - 2b_2)y_{1i,j-1} + (\alpha + 2\beta)\xi_{1i,j},$$

(7.13)

$$y_{2i,j} = a_1 y_{1,i-1,j} + a_2 y_{1i,j-1} + \alpha\xi_{2i,j}$$

. (7.4)

Авторегрессионные модели типа Хабиби

Векторным аналогом скалярной КФ

$V_x(i, j) = M[x_{ij}x_{00}] = \delta^2 \rho_1^{|i|}\rho_2^{|j|}$ модели (3.4) является КФ

$$V_x(i, j) = M[\bar{x}_{ij}\bar{x}_{00}^T] = A_1^{i+}A_2^{j+}VA_1^{|i-|}A_2^{|j-|},$$

(7.15)

где A_1 и A_2 – перестановочные между собой матрицы, все собственные числа которых меньше единицы; $V=V_x(0,0)$; $i^+ = \max(0,i)$, $i^- = \max(0,-i)$. Матрицы A_1 и A_2 являются своего рода матричными коэффициентами корреляции на единичном расстоянии по соответствующим осям координат.

Векторное СП с КФ (7.15) может быть задано на квадранте с помощью авторегрессионных уравнений типа модели Хабиби:

$$\begin{aligned}
\bar{x}_{00} &= W\bar{\xi}_{00}, \\
\bar{x}_{i0} &= A_1\bar{x}_{i-1,0} + U_1\bar{\xi}_{i0}, && i \geq 1, \\
\bar{x}_{0j} &= A_2\bar{x}_{0,j-1} + U_2\bar{\xi}_{0j}, && j \geq 1, \\
\bar{x}_{ij} &= A_1\bar{x}_{i-1,j} + A_2\bar{x}_{i,j-1} - A_1A_2\bar{x}_{i-1,j-1} + U\bar{\xi}_{ij}, && i \geq 1, j \geq 1,
\end{aligned}$$

(7.16)

где матрицы W, U_1, U_2 и U должны удовлетворять условиям

$WW^T = V$, $U_1U_1^T = V - A_1VA_1^T$,

$UU^T = V - A_1VA_1^T - A_2VA_2^T + (A_1A_2)V(A_1A_2)^T$,

$U_2U_2^T = V - A_2VA_2^T$, для получения КФ (7.15). Модель (7.16) легко обобщается на многомерный случай с КФ, аналогичной (7.15).

6.7.2. Разложимые векторные случайные поля

Векторное СП с КФ, определяемой формулой (7.9), можно получить другим способом. Рассмотрим m независимых скалярных СП (или векторное СП с независимыми компонентами)

$$(y_{1i}, y_{2i}, \ldots, y_{mi})^T = \bar{y}_i,$$

(7.17)

где СП y_{ki} имеет КФ $V_1(i)$, а остальные компоненты имеют КФ $V_2(i)$. Построим векторное СП $\bar{x}_i$ как линейное преобразование вида

$$\begin{pmatrix} x_{1t} \\ x_{2t} \\ x_{3t} \\ \dots \\ x_{mt} \end{pmatrix} = \begin{pmatrix} \sqrt{\frac{1}{m}} & -\sqrt{\frac{1}{1\cdot 2}} & -\sqrt{\frac{1}{2\cdot 3}} & -\sqrt{\frac{1}{3\cdot 4}} & \dots & -\sqrt{\frac{1}{(m-1)\cdot m}} \\ \sqrt{\frac{1}{m}} & \sqrt{\frac{1}{2}} & -\sqrt{\frac{1}{2\cdot 3}} & -\sqrt{\frac{1}{3\cdot 4}} & \dots & -\sqrt{\frac{1}{(m-1)\cdot m}} \\ \sqrt{\frac{1}{m}} & 0 & \sqrt{\frac{2}{3}} & -\sqrt{\frac{1}{3\cdot 4}} & \dots & -\sqrt{\frac{1}{(m-1)\cdot m}} \\ \dots & \dots & \dots & \dots & \dots & \dots \\ \sqrt{\frac{1}{m}} & 0 & 0 & 0 & \dots & \sqrt{\frac{m-1}{m}} \end{pmatrix} \begin{pmatrix} y_{1t} \\ y_{2t} \\ y_{3t} \\ \dots \\ y_{mt} \end{pmatrix} \tag{7.18}$$

или в сжатом виде

$$\vec{x}_t = G\vec{y}_t \,. \tag{7.19}$$

Построенное поле $\vec{x}_t$ имеет КФ

$$V_x(i) = M[\vec{x}_t \vec{x}_t^T] = M[G\vec{y}_t \vec{y}_t^T G^T] = GM[\vec{y}_t \vec{y}_t^T]G^T = GV_y(i)G^T = G \begin{pmatrix} V_1(i) & 0 & 0 & \dots & 0 \\ 0 & V_2(i) & 0 & \dots & 0 \\ 0 & 0 & V_3(i) & \dots & 0 \\ \dots & \dots & \dots & \dots & \dots \\ 0 & 0 & 0 & \dots & V_m(i) \end{pmatrix} G^T$$

что, как нетрудно убедиться, совпадает с (7.9).

При этом G не зависит от $V_1(i)$ и $V_2(i)$. Кроме того, $G^{-1} = G^T$ (G – ортогональная матрица). Следовательно, $\vec{y}_t = G^{-1}\vec{x}_t = G^T\vec{x}_t$, т. е. преобразование $\vec{y}_t = G^T\vec{x}_t$ является **декоррелирующим** для $\vec{x}_t$: результат его применения $\vec{y}_t$ состоит из некоррелированных СВ.

Представление (7.19) можно обобщить, взяв в нем произвольную матрицу G и $\vec{y}_t = (y_{1t}, \dots, y_{mt})$, состоящее из m некоррелированных между собой скалярных полей с КФ $V_1(i)$,…, $V_m(i)$. Назовем такие поля **разложимыми**.

Если составляющие y_{kt} однородны, то получаемое векторное СП также однородно и имеет КФ $V_x(i) = GV_y(i)G^T$, где $V_y(i) = diag(V_1(i), \dots, V_m(i))$ – клеточно-диагональная матрица.

Класс разложимых векторных СП можно расширить, если в (7.19) использовать переменную матрицу преобразования:

$$\vec{x}_t = G_t\vec{y}_t \,. \tag{7.20}$$

В этом случае поле $\vec{x}_t$ имеет КФ

$$V_x(t, \tau) = G_t V_y(t, \tau) G_t^T \,. \tag{7.21}$$

Для описания образующих скалярных полей y_{kt} в моделях (7.19) и (7.20) могут быть использованы любые модели скалярных СП.

6.7.3. Волновые модели векторных случайных полей

Волновые модели скалярных СП могут быть обобщены на случай векторных полей. Наиболее простым частным случаем такой модели является модель с волнами вида

$$f(U,t),(\vec{x}_k,\tau_k),\vec{w}_k) = g(U,t),(\vec{x}_k,\tau_k),\vec{w}_k)G\vec{\xi}_k$$

, (7.22)

где g(*) – скалярная функция; G – матрица; векторы $\vec{\xi}_k$ независимы между собой и имеют независимые стандартные компоненты. В этом случае векторы одной волны коллинеарны случайному вектору $G\vec{\xi}_k$, т. е. возмущающее влияние каждой волны состоит в прибавлении к имеющимся векторам поля однонаправленных векторов. Такое поле имеет КФ вида

$$V_f(*) = GV_f(*)G^T$$

, (7.23)

где $V_f(*)$ – КФ скалярного волнового поля, порождаемого функцией (7.22) при замене в ней $G\vec{\xi}_k$ на стандартные скалярные ξ_k с тем же совместным распределением, что и у компонент вектора $\vec{\xi}_k$.

Более сложные модели можно получить, взяв в (7.22) векторную функцию

$$f(*) = G(U,t),(\vec{x}_k,\tau_k),\vec{w}_k)\xi_k$$

, (7.24)

где G(*) – случайный или переменный матричный коэффициент. В этом случае каждая волна может состоять из векторов, направленных по-разному в различных точках пространства, например, в виде вихря. В результате порождаемые векторные поля могут иметь более сложную структуру и описыать более широкий спектр реальных случайных векторных полей. Существенно, что **эта модель описывает последовательность кадров, что позволяет моделировать динамические векторные случайные процессы, например, параметры переменных геометрических искажений многомерных И** или поле скоростей движения жидкостей и газов.

6.7.4. Тензорные модели векторных и более сложнозначных случайных полей

Тензорная модель (5.1) или (5.2) пригодна для описания последовательностей скалярных кадров, определенных на сетке W любой размерности. Для представления последовательности векторнозначных кадров можно использовать ту же модель, увеличив размерность W на единицу. Эта дополнительная размерность используется для компонент векторов моделируемого векторного СП. **Такая модель может быть использована, например, для описания последовательности цветных И. В этом случае модель определяет вектор цвета в каждом элементе И.**

Аналогичный прием может быть использован для построения моделей и более сложных полей, значениями которых являются m-мерные матрицы или тензоры ранга m. Для этого размерность сетки W увеличивается на m единиц. Например, таким способом можно описать СП тензоров деформаций (**геометрических межкадровых искажений И**) или напряжений, изменяющихся со временем.

7. СТАТИСТИЧЕСКИЕ РЕШЕНИЯ

Человеческому обществу и всем его частям, вплоть до отдельных личностей, в процессе своей жизнедеятельности приходится постоянно выполнять различные действия, направленные на достижение определенных целей. Человек разумный отличается способностью планирования своих действий, т. е. он сначала принимает решение (желательно, хорошее), выбирая его из множества возможных. Искусство принятия хороших решений приходит с опытом, и немаловажен здесь природный дар. Можно с уверенностью сказать, что все выдающиеся деятели (ученые, изобретатели, политики, военачальники и т. д.) достигли своих успехов именно благодаря умению принимать правильные решения в своей области деятельности.

Задачи обработки МИ тоже могут быть сформулированы как задачи принятия решений СИИ. Например, при обнаружении объектов на фоне МИ СИИ должна выбрать одно из решений: «есть объект» или

«нет объекта», а при оценке параметров МИ СИИ должна выбрать какой-то набор их возможных значений.

Следует иметь математический аппарат для нахождения оптимальных решений СИИ. Для ряда ситуаций таким аппаратом является **теория статистических решений**, основные положения и результаты которой приводятся в этой главе.

7.1. Решения

Решение – это результат целенаправленной обработки имеющейся информации. Приведем несколько примеров. Решение может заключаться в выборе одного из действий в бизнесе, политике, лечении больного и т. д. **Оценка параметра – тоже решение, состоящее в выборе какого-то числа из массы возможных значений**. **В задаче обнаружения выбирается одно из двух решений – есть объект или его нет. В задаче распознавания принимаются решения о принадлежности исследуемого объекта к одному из возможных классов.**

Решение, таким образом, является очень общим понятием. Любая задача подразумевает принятие некоторого решения.

Отметим некоторые особенности, которые следует учитывать СИИ, принимая решения.

1^0. Решение СИИ направлено на достижение некоторой **цели** и приводит к некоторым **последствиям**, по которым должно оцениваться качество решения.

2^0. Решение СИИ выбирается из **множества альтернатив** (возможных решений), конечного или бесконечного. При построении математической теории множество альтернатив предполагается **определенным**. Это предположение не ограничивает общности теории, так как новые, нестандартные решения СИИ можно включить в множество уже имеющихся альтернатив.

3^0. Решение СИИ принимается по доступной к моменту его принятия **информации**, состоящей из двух принципиально различных частей. Первая ее часть обобщает весь прошлый опыт, т. е. **априорна**. Вторая часть – совокупность данных наблюдения – получается непосредственно в процессе выработки решений. Эта **апостериорная** совокупность данных является объектом обработки в процессе принятия решения СИИ. Например, это может быть наблюдаемым изображением, набором анализов, состоянием биржи и т. д.

В практических ситуациях соотношение объемов этих двух частей информации может меняться в широких пределах. В частности, одна из них может полностью отсутствовать.

4^0. Очень часто доступная информация и последствия от принятия решения имеют **статистическую природу** из-за ненаблюдаемости скрытых случайных факторов. Поэтому и процедура принятия решений СИИ должна иметь статистический характер. Именно такой случай рассматривается в теории статистических решений.

5^0. Решение СИИ часто заключается в принятии совокупности частичных решений, т. е. **решение может быть векторным или даже многомерным.** Например, в различных задачах обработки изображения результатом является совокупность частных решений, касающихся отдельных элементов наблюдаемого изображения.

6^0. Выбор решения СИИ из множества альтернатив не всегда однозначно определяется имеющейся информацией. Он может (а иногда и должен) допускать элементы случайности. Это так называемые **рандомизированные решения**. Такие решения используются, например, в теории игр. И в реальной жизни мы иногда принимаем решение, бросая монетку.

Задачей является выбор такого решения СИИ, которое приводило бы к наиболее благоприятным последствиям. Соответствующие **правила принятия решений** называются **оптимальными**. Для их нахождения разработан и постоянно совершенствуется мощный математический аппарат (теория игр; линейное, нелинейное и динамическое программирование; теория статистических решений и т.

д.), позволяющий рассматривать очень многообразные задачи с общих математических позиций.

Таким образом, решения СИИ выбираются из множества альтернатив $U = \{u\}$. Решение СИИ выбирается с использованием **данных (наблюдений, измерений, выборки)** z, всевозможные значения которых составляют множество $Z = \{z\}$. При этом имеются некоторые скрытые, ненаблюдаемые параметры θ, описывающие реальную ситуацию и принимающие значения из множества $\Theta = \{\theta\}$ всех возможных ситуаций.

Множество альтернатив U может быть самым разнообразным (совокупность оценок параметров, управлений, и прочих акций). **Математически же можно выделить три случая**.

а) $U = \{u_1, u_2, \dots\}$, т. е. пространство решений СИИ дискретно (конечно или счетно). Например, в задачах обнаружения и распознавания объектов.

б) Пространство решений СИИ непрерывно (ограничено или бесконечно). Например, в задаче оценки параметров.

в) Пространство дискретно–непрерывно: $U = \{U_1, U_2, \dots, U_K\}$, где все или некоторые из U_i непрерывны. Например, задача обнаружения объектов с оценкой их параметров (координат, скорости и т. д.).

Правила принятия решений СИИ могут быть нерандомизированными и рандомизированными.

Нерандомизированные правила определяют для каждого наблюдения z вполне определенное решение u. Такие правила имеют вид $u = u(z)$, т. е. являются **функциями, преобразованиями** $z \to u$.Совокупность различных преобразований

$u(z)$ (любых или из некоторого ограниченного класса) образует множество $U(z)$ всех нерандомизированных решающих правил, из которого СИИ и следует выбирать оптимальное правило.

Отметим, что U и $U(z)$ – разные множества. Например, в задаче обнаружения объекта $U = \{H_0, H_1\}$, где $H_0 =$ (нет объекта) и $H_1 =$ (есть объект), а множество $U(z)$ – это набор всевозможных правил $u(z)$, которые каждому возможному наблюдению z ставят в соответствие определенное решение H_0 или H_1. Геометрически это можно представить в виде рис.7.1, на котором пространство Z условно представлено прямоугольником. Пусть $u = u(z)$ – некоторое решающее правило, которое на одной части Z_0 наблюдений принимает значение H_0, а на остальной части Z_1 – значение H_1. Таким образом, $U(z)$ геометрически есть набор всевозможных разбиений Z на два подпространства Z_0 и Z_1.

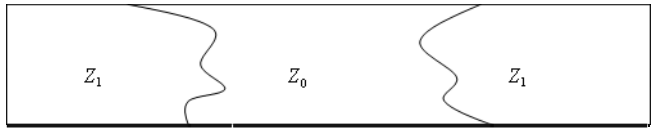


Рис.7.1.

Рандомизированные правила определяются заданием на U условной вероятностной меры $\Phi(u \mid z)$, т. е. при имеющемся

наблюдении z_0 решение u выбирается случайным образом с распределением вероятностей $\Phi(u \mid z_0)$. Нерандомизированное правило $u(z)$ есть частный, вырожденный случай рандомизированного, когда мера $\Phi(u \mid z)$ целиком сосредоточена в точке $u(z)$.

7.2. Потери и риск

Последствия от принятия решений СИИ следует оценивать по степени их соответствия поставленной цели. Это соответствие, в принципе, может быть определено с помощью количественной меры, определяющей выигрыш или потери от принятия решения СИИ. Эта мера называется **функцией потерь (штрафа)** или **функцией выигрыша (целевой функцией)**. Для определенности будем использовать функцию потерь, которую следует **минимизировать**, чтобы решение СИИ было оптимальным.

Потери могут зависеть не только от принятого СИИ решения u, но и от реальной ситуации, описываемой параметрами θ. Возможно, что потери зависят и от наблюдений z. В общем случае функция потерь имеет вид $g = g(u, \theta, z)$. В частных случаях может быть $g = g(u, \theta)$, $g = g(u, \theta_0) = g(u)$ и т. д.

Если бы значение θ было известно, т. е. полностью определена реальная ситуация θ_0, то при получении наблюдения $z = z_0$ оптимальным решением была бы точка минимума u_0 функции $g(u) = g(u, \theta_0, z_0)$, т. е. **оптимальным решающим правилом** была бы просто минимизация известной функции по u при заданных

значениях ее параметров θ и z. Полученное таким образом решение является каждый раз наилучшим.

В реальности же θ являются скрытыми параметрами, т. е. действительная обстановка может быть, в лучшем случае, известна только приближенно. В таком случае уже невозможно каждый раз находить наилучшее решение – неизбежны просчеты из-за ошибочной оценки ситуации.

Единственное, что можно сделать в сложившемся положении, это попытаться найти такое решающее правило, при котором минимальными будут **средние потери**, т. е. среднее значение функции потерь, называемое **риском**. **Правило, минимизирующее риск, будем называть оптимальным.**

Для нахождения среднего значения функции потерь $g = g(u, \theta, z)$ нужно задать распределение вероятностей на пространстве ее аргументов U, Θ, Z, что может быть сделано следующим образом.

Закон распределения вероятностей возможных ситуаций θ описывается ПРВ $W(\theta)$ на Θ.

Наблюдения z должны быть в той или иной мере связаны с θ, иначе в них нет никакой надобности. Эта связь может быть описана условной ПРВ $P(z \mid \theta)$ на пространстве наблюдений Z, которая называется **функцией правдоподобия** (ФП).

Решения u принимаются в зависимости от z, это и есть решающее правило. В наиболее общем случае это правило рандомизировано и определяется **условной вероятностной мерой**, для определенности заданной условной ПРВ $\Phi(u \mid z)$.

Таким образом, совместная ПРВ параметров u, θ, z есть функция $W(u,\theta,z) = W(\theta)P(z \mid \theta)\varphi(u \mid z)$, определенная на пространстве $U \times \Theta \times Z$.

В зависимости от степени усреднения можно рассматривать различные типы рисков.

Средний риск. Если известно распределение θ и z (заданы ПРВ $W(\theta)$ и $P(z \mid \theta)$), то для любого решающего правила $\varphi = \varphi(u \mid z)$ можно найти безусловное математическое ожидание функции потерь, зависящее только от этого правила и называемое **средним риском:**

$$R(\varphi) = M[g(u,\theta,z)] = \int_{\theta}\int_{z}\int_{U} g(u,\theta,z)\varphi(u \mid z)P(z \mid \theta)W(\theta)\,du\,dz\,d\theta \quad (7.1)$$

Оптимизация принятия решений заключается в выборе такого правила (т. е. ПРВ $\varphi(u \mid z)$), при котором средний риск (7.1) минимален.

Для нерандомизированых правил $u = u(z)$ распределение $\varphi(u \mid z)$ сосредоточено в точке $u(z)$, т. е.

$$\varphi(u \mid z) = \delta(u - u(z)), \quad (7.2)$$

где $\delta(\cdot)$ – функция Дирака, и (7.1) принимает вид

$$R(u(z)) = \int\int g(u(z),\theta,z)P(z \mid \theta)W(\theta)\,dz\,d\theta \quad (7.3)$$

Здесь и в дальнейшем, если не возникает недоразумений, будем опускать обозначения областей интегрирования. Выражение (7.3) определяет средний риск для любого правила $u = u(z)$, а оптимизация принятия решений СИИ заключается в выборе такого правила $u(z)$, при котором риск (7.3) минимален.

Априорным риском называется величина

$$G(u) = \int\int g(u,\theta,z)P(z \mid \theta)W(\theta)\,dz\,d\theta, \quad (7.4)$$

где u не зависит от z. Если, кроме того, $g(u,\theta,z) = g(u,\theta)$, то

$$G(u) = \int\int g(u,\theta)P(z \mid \theta)W(\theta)\,dz\,d\theta = \int g(u,\theta)W(\theta)\,d\theta. \quad (7.5)$$

Выражения (7.4) и (7.5) определяют **априорную оценку потерь**, связанных с решением u, принимаемым при отсутствии наблюдений или при их игнорировании. Такое решение СИИ может быть спланировано заранее, еще до наступления момента, когда решение необходимо принимать. В такой постановке оптимальным решением СИИ является u^0, минимизирующее (7.4) или (7.5), оно учитывает только априорную информацию.

Апостериорным риском называется условное математическое ожидание функции потерь для данного решения u при данном значении z. Для этого найдем по формуле Байеса апостериорное условное распределение θ при заданном значении z:

$$W_a(\theta) = W(\theta \mid z) = \frac{P(z \mid \theta)W(\theta)}{\int P(z \mid \theta)W(\theta)d\theta}.$$ (7.6)

Апостериорный риск определится усреднением функции потерь по ПРВ (7.6):

$$R(u,z) = \int g(u,\theta,z)W(\theta \mid z)d\theta = \frac{\int g(u,\theta,z)P(z \mid \theta)W(\theta)d\theta}{\int P(z \mid \theta)W(\theta)d\theta}.$$ (7.7)

Средний и апостериорный риски связаны соотношением

$$R(\varphi) = \iint R(u,z)\varphi(u \mid z)P(z)dudz,$$ (7.8)

где

$$P(z) = \int P(z \mid \theta)W(\theta)d\theta -$$ (7.9)

безусловная ПРВ наблюдений z. В случае нерандомизированных правил соотношение (7.8) принимает вид

$$R(u(z)) = \int R(u(z),z)P(z)dz.$$ (7.10)

Условным риском называются средние потери при использовании правила $\varphi = \varphi(u \mid z)$ при заданном значении θ:

$$r(\varphi,\theta) = \iint g(u,\theta,z)\varphi(u \mid z)P(z \mid \theta)dudz,$$ (7.11)

а в случае нерандомизированного правила $u = u(z)$:

$$r(u(z),\theta) = \int g(u(z),\theta,z)P(z \mid \theta)dz.$$ (7.12)

Этот риск определяет потери в среднем по всем возможным наблюдениям для конкретного решающего правила $u = u(z)$ и конкретной ситуации θ.

7.3. Байесовы решающие правила

Если известны распределения $W(\theta)$ и $P(z \mid \theta)$, т. е. если имеется статистическое описание параметров θ и наблюдений z, то, в принципе, задача нахождения оптимального решающего правила легко решается (остаются только некоторые вычислительные проблемы). Само оптимальное решающее правило в этих условиях называется **байесовым**. Выведем это правило в общем случае.

Оптимальное решающее правило $\varphi = \varphi(u \mid z)$ должно минимизировать средний риск (7.8). В силу неотрицательности ПРВ $P(z)$ минимальное значение интеграла в (7.8) достигается, если при любом z минимален интеграл

$$\int P(u,z)\varphi(u \mid z)du.$$

Этот интеграл в силу неотрицательности $\varphi(u \mid z)$ минимален, когда распределение $\varphi(u \mid z)$ целиком сосредоточено в точке $u = u_0 = u_0(z)$ минимума апостериорного риска $R(u,z)$, т. е. при

$$\varphi(u \mid z) = \delta(u - u_0(z)).$$

Следовательно, байесово правило является нерандомизированным и при каждом наблюдении z оптимальным является решение, минимизирующее апостериорный риск. Этого и следовало ожидать, так как **апостериорный риск – это риск при известном наблюдении**.

Таким образом, задача построения оптимального решающего правила сводится к относительно простой задаче минимизации функции $R(u,z)$ на множестве альтернатив U при заданном наблюдении z.

Байесово решение $u_0(z)$ определяется из соотношения

$$R(u_0(z),z) = \min_u \int g(u,\theta,z)W(\theta|z)d\theta, \quad (7.13)$$

что (с учетом независимости знаменателя в (7.6) и (7.7) от u) эквивалентно соотношению

$$\int g(u_0(z),\theta,z)P(z|\theta)W(\theta)d\theta = \min_u \int g(u,\theta,z)P(z|\theta)W(\theta)d\theta. \quad (2.14)$$

Получаемый при этом минимальный средний риск

$$R^* = R(u_0(z)) = \int (\min_u \int g(u,\theta,z)P(z|\theta)W(\theta)d\theta)dz \quad (7.15)$$

называется **байесовым**, его обеспечивает само байесово решение.

7.4. Многоальтернативные решения

Рассмотрим важный частный случай решения, заключающегося в выборе одной из m альтернатив $u = i = 0,1,...,m-1$, т. е. $U = \{0,1,...,m-1\}$. При этом $\Theta = \{j = 0,1,...,n-1\}$, т. е. возможна одна из n ситуаций $\theta = j = 0,1,...,n-1$. Этой схеме

соответствуют задачи проверки гипотез, обнаружения и различения сигналов, распознавания образов и т. д. Пусть функция потерь не зависит от z и

$$g(u,\theta,z) = g(u=i,\theta=j) = g_{ij},$$

т. е. функция потерь задана $m \times n$**-матрицей потерь** $\{g_{ij}\}$. Тогда

$$R(u,z) = R(u=i,z) = \sum_j g_{ij}p(\theta=j|z) = \sum_j g_{ij}\frac{P(z|\theta=j)p(\theta=j)}{P(z)} =$$

$$= \frac{1}{P(z)}\sum_j g_{ij}p(\theta=j)P(z|\theta=j) = \frac{1}{P(z)}\sum_j a_{ij}P(z|\theta=j) = \frac{1}{P(z)}L_i, \quad (2.16)$$

где $p(\theta=j)$ – априорные вероятности ситуаций; $P(z|\theta=j)$ – ФП; $P(z) = \sum_j p(\theta=j)P(z|\theta=j)$ – вероятность наблюдения z; $p(\theta=j|z)$ – условная вероятность ситуации j при наблюдении z; $a_{ij} = g_{ij}p(\theta=j)$.

Из (7.16) следует, что при имеющемся наблюдении z нужно выбрать такое решение $u=i$, при котором минимальна линейная комбинация L_i ФП $P(z|\theta=j)$ с коэффициентами a_{ij}, т. е. нужно сравнить между собой m линейных комбинаций $L_0,...,L_{m-1}$.

Отметим, что коэффициенты a_{ij} этих линейных комбинаций зависят от функции потерь g_{ij} и априорных вероятностей $p(\theta=j)$

ситуаций, но от наблюдений не зависят, т. е. концентрируют в себе априорную информацию. Значения же $P(z \mid \theta = j)$, входящие в L_i, напротив, зависят от наблюдений.

В системе ИИ определяющую роль в принятии решения должны играть именно наблюдения. Если это не так, то решение будет приниматься в основном по априорной информации, а сама информационная система окажется практически бесполезной. В этом заключается общая закономерность систем обработки информации: чем более высокими качествами должна обладать информационная система, тем меньшее значение имеют априорные данные о характеристиках потерь и поведении параметров θ.

Для рассматриваемой задачи это означает, что основное значение должен иметь разброс значений $P(z \mid \theta = 0), \ldots, P(z \mid \theta = m-1)$, а не разброс коэффициентов $a_{i_1}, \ldots, a_{i_{n-1}}$. А именно, существенно большим должно быть значение $P(z \mid \theta = j_0)$, соответствующее действительно имеющей место ситуации j_0. Другими словами, наблюдения в хорошей информационной системе должны достаточно точно идентифицировать имеющуюся ситуацию. В этом случае априорные сведения имеют очень малое влияние, поэтому их можно выбирать практически произвольно.

Но это все, конечно, только пожелания о качествах системы обработки информации. В действительности приходится работать с той системой, какая есть. В любом случае оптимальное решение соответствует минимальной из линейных комбинаций L_i.

Двухальтернативные решения

Рассмотрим частный случай двухальтернативных задач, когда $m = n = 2$. Этому случаю соответствует, например, задача обнаружения сигналов или других объектов, в применении к которой и произведем все выкладки.

Итак, возможны две ситуации или гипотезы: H_0 – нет сигнала и H_1 – есть сигнал. Решение состоит в выборе одной из этих гипотез. Заданы априорные вероятности $p(H_0)$ и $p(H_1)$ и функция потерь $g_{ij} = g(u = H_i, \theta = H_j)$. При этом g_{01} и g_{10} – потери при неверных решениях, а g_{00} и g_{11} – потери при верных решениях. Задана также ФП: $P(z \mid H_0)$ – распределение вероятностей наблюдений при отсутствии сигнала и $P(z \mid H_1)$ – при его наличии.

Из (7.16) следует, что решение $u = H_1$ принимается при выполнении неравенства $L_1 < L_0$, т. е. если

$$a_{10}P(z \mid H_0) + a_{11}P(z \mid H_1) < a_{00}P(z \mid H_0) + a_{01}P(z \mid H_1),$$

или в эквивалентном виде

$$(a_{01} - a_{11})P(z \mid H_1) > (a_{10} - a_{00})P(z \mid H_0),$$

$$(g_{01} - g_{11})p(H_1)P(z \mid H_1) > (g_{10} - g_{00})p(H_0)P(z \mid H_0).$$

Естественно, что $g_{01} > g_{11}$ и $g_{10} > g_{00}$ (потери при верном решении должны быть меньше, чем при ошибочном). Поэтому решающее правило принимает вид

$$\Lambda(z) = \frac{p(z \mid H_1)}{p(z \mid H_0)} \begin{cases} > \Lambda_0 \Rightarrow H_1, \\ \leq \Lambda_0 \Rightarrow H_0, \end{cases} \qquad (7.17)$$

где

$$\Lambda_0 = \frac{g_{10} - g_{00}}{g_{01} - g_{11}} \frac{p(H_0)}{p(H_1)}, \qquad (7.18)$$

пороговое значение (порог) решающего правила.

Отношение $\Lambda(z) = p(z \mid H_1) / p(z \mid H_0)$ называется **отношением правдоподобия** (ОП). Оказывается, что ОП является **достаточной статистикой** для рассматриваемой задачи, т. е. вся информация, содержащаяся в наблюдениях z, **сконцентрирована в единственном числе – значении ОП**. Это значение нужно сравнить с порогом Λ_0, который зависит от априорной вероятности появления сигнала $p(H_1)$ (отметим, что $p(H_0) = 1 - p(H_1)$) и от функции потерь g_{ij}, т. е. от критерия оптимальности обнаружения. Если выбрать какой-то другой критерий оптимальности, то сменится только значение порога Λ_0, а правило обнаружения сохранит вид (7.17).

Отметим, что в общем случае (не обязательно для задачи обнаружения) правило (7.17) имеет вид

$$\Lambda(z) = \frac{p(z \mid \theta = 1)}{p(z \mid \theta = 0)} \begin{cases} > \Lambda_0 \Rightarrow u = 1, \\ \le \Lambda_0 \Rightarrow u = 0. \end{cases} \qquad (7.19)$$

7.5. Оценка параметров. Методы построения оценок

Вторым важнейшим частным случаем статистических решений является оценка параметров.

Пусть искомое решение заключается в построении оценки $u = (u_1, \ldots, u_m)$ векторного параметра $\theta = (\theta_1, \ldots, \theta_n)$ по совокупности наблюдений z, где каждая из компонент имеет неограниченную область значений. Этот случай охватывает многочисленные задачи прогноза, фильтрации, оценки характеристик МИ и т. д.

Оптимальное решающее правило (оптимальная оценка) и в этом случае определяется соотношением (7.13). Однако при некоторых естественных предположениях можно получить правила в более простых формах.

Предположим, что функция потерь не зависит от z и симметрична относительно ошибки оценки: $g(u, \theta, z) = g(|\theta - u|)$. Например, при **квадратичной функции потерь** $g(t) = t^2$ получаем **метод наименьших квадратов** (МНК): в качестве оценки выбирается такое значение u, при котором среднее значение квадрата ошибки оценки $(\theta - u)^2$ минимально (если оценка несмещенная, то минимальна дисперсия ошибки). Если взять $g(t) = |t|$, то получим метод **минимального среднего модуля ошибки** и т. д. При разных функциях потерь, т. е. при разных понятиях оптимальности, могут получаться и разные оптимальные оценки. Например, при $g(t) = t^2$ оптимальной оценкой является (условное) математическое ожидание параметра θ, а при $g(t) = |t|$ его (условная) медиана. Это разнообразие оптимальных (каждая в своем смысле) оценок нежелательно.

Предположим дополнительно, что апостериорная ПРВ $W(\theta \mid z)$ хотя бы приблизительно симметрична относительно некоторой точки $\hat{\theta}(z)$, зависящей от z, и что

$$\lim_{\theta\to\infty} g(\theta-\hat{\theta}(z))W(\theta\mid z)=0$$

,

т. е. что функция потерь на бесконечности возрастает не слишком быстро. При этих условиях из (7.12) следует, что оптимальное решение u, т. е. оптимальная оценка неизвестного θ, определяется как

$$u=u_0(z)=\hat{\theta}(z) \qquad (7.20)$$

независимо от конкретного вида функции потерь $g(\cdot)$, функции правдоподобия $P(z\mid\theta)$ и априорного распределения $W(\theta)$.

Точка $\hat{\theta}(z)$ обладает тем свойством, что в ней достигается максимум апостериорной ПРВ $W(\theta\mid z)$, что и дает универсальный способ нахождения оптимальных оценок – **метод максимума апостериорной ПРВ** (МАП): оптимальной оценкой $\hat{\theta}(z)$ параметра θ при наблюдении z является точка максимума апостериорной ПРВ $W(\theta\mid z)$ по θ:

$$W(\hat{\theta}(z)\mid z)=\max_{\theta} W(\theta\mid z) \qquad (7.21)$$

.

Как и в п.7.4, при высокой информативности наблюдений априорные сведения должны слабо влиять на вид оптимального решения. При этом предположении можно получить другое решающее правило. Для этого представим (7.21) в эквивалентном виде

$$P(z\mid\hat{\theta}(z))W(\hat{\theta}(z))=\max_{\theta} P(z\mid\theta)W(\theta) \qquad (7.22)$$

.

Если априорная информация (т. е. ПРВ $W(\theta)$) мало влияет на решение, то определяющей должна быть ФП $P(z\mid\theta)$, поэтому, заменяя (7.22) на приближенное уравнение

$$P(z\mid\hat{\theta}(z))=\max_{\theta} P(z\mid\theta) \qquad (7.23)$$

,

получаем так называемый **метод максимального правдоподобия** (ММП): в качестве оценки $\hat{\theta}=\hat{\theta}(z)$ берется точка максимума ФП $P(z\mid\theta)$.

Замечание. ММП – все же приближенный метод, в нем априорная информация полностью игнорируется, поэтому иногда получаемые с помощью этого метода оценки существенно хуже оценок МАП.

7.6. Оценка гауссовских параметров по гауссовским наблюдениям

Во многих приложениях данные имеют гауссовские распределения, поэтому рассмотрим этот случай подробнее.

Совместная ПРВ n гауссовских СВ $y_1,\dots,y_n$, составляющих гауссовский вектор $\bar{y}=(y_1,\dots,y_n)^T$, имеет вид

$$w(y_1,\dots,y_n)=w(\bar{y})=\frac{1}{(2\pi)^{n/2}\det^{1/2}(V)}\exp\left(-\frac{1}{2}(\bar{y}-\bar{m})^T V^{-1}(\bar{y}-\bar{m})\right) \qquad (7.24)$$

,

где

$$\bar{m}=M[\bar{y}]=(M[y_1],M[y_2],\dots,M[y_n])^T=(m_1,\dots,m_n)^T$$

– вектор математических ожиданий (математическое ожидание вектора $\bar{y}$);

$V = M[(\bar{y} - \bar{m})(\bar{y} - \bar{m})^T] = \left(M[(\bar{y}_i - \bar{m}_i)(\bar{y}_j - \bar{m}_j)^T]\right)_{i,j=\overline{1,n}}$ – матрица ковариаций. Отметим, что V – симметричная матрица: $V^T=V$.

В частности, если $\bar{m} = 0$, т. е. $M[y_i] = 0$, то

$$w(\bar{y}) = \frac{1}{(2\pi)^{n/2}\det^{1/2}(V)}\exp\left(-\frac{1}{2}\bar{y}^T V^{-1}\bar{y}\right), \quad (7.25)$$

где $V = M[\bar{y}\bar{y}^T]$.

Оптимальная оценка

Рассмотрим задачу оценки гауссовского вектора $\bar{x} = (x_1,\dots,x_m)^T$, когда имеется гауссовский вектор наблюдений $\bar{z} = (z_1,\dots,z_n)^T$. Пусть известны все средние значения и все ковариации СВ x_i, $i = \overline{1,m}$, и z_i, $i = \overline{1,n}$. Без потери общности можно считать, что $M[\bar{x}] = 0$ и $M[\bar{z}] = 0$, так как мы можем центрировать все СВ, вычитая из них их математические ожидания. Будем искать оптимальные оценки в смысле минимума средних квадратов ошибок, т. е. при квадратичной функции потерь

$$M[(\hat{x}_i - x_i)^2] = \min. \quad (7.26)$$

По общей теории статистических решений (п.7.5), оптимальная оценка $\hat{\bar{x}}$ в рассматриваемом случае есть оценка по методу МАП:

$$w(\hat{\bar{x}}, \bar{z}) = \max_{\bar{x}} w(\bar{x}, \bar{z}). \quad (7.27)$$

Представим $w(\hat{\bar{x}}, \bar{z})$ в виде (7.25). Для этого объединим $\bar{x}$ и $\bar{z}$ в один вектор $\bar{y} = \begin{pmatrix}\bar{x}\\ \bar{z}\end{pmatrix}$, тогда

$$V = \begin{pmatrix} V_{xx} & V_{xz} \\ V_{zx} & V_{zz} \end{pmatrix}, \quad (7.28)$$

где $V_{xx} = M[\bar{x}\bar{x}^T]$; $V_{zz} = M[\bar{z}\bar{z}^T]$; $V_{xz} = M[\bar{x}\bar{z}^T]$; $V_{zx} = M[\bar{z}\bar{x}^T]$; $V_{xz}^T = V_{zx}$. Таким образом,

$$w(\bar{x}, \bar{z}) = \frac{1}{(2\pi)^{m+n/2}\det^{1/2}(V)}\exp\left(-\frac{1}{2}(\bar{x}^T, \bar{z}^T)\begin{pmatrix} V_{xx} & V_{xz} \\ V_{zx} & V_{zz} \end{pmatrix}^{-1}\begin{pmatrix}\bar{x}\\ \bar{z}\end{pmatrix}\right). \quad (7.29)$$

Максимум (7.29) достигается при минимальном значении выражения

$$(\bar{x}^T, \bar{z}^T)\begin{pmatrix} V_{xx} & V_{xz} \\ V_{zx} & V_{zz} \end{pmatrix}^{-1}\begin{pmatrix}\bar{x}\\ \bar{z}\end{pmatrix}. \quad (7.30)$$

Пусть

$$\begin{pmatrix} V_{xx} & V_{xz} \\ V_{zx} & V_{zz} \end{pmatrix}^{-1} = \begin{pmatrix} K & L \\ M & N \end{pmatrix}, \quad (7.31)$$

где K, L, M, N – матрицы размеров $m \times m, m \times n, n \times m, n \times n$, соответственно, и M=LT. Тогда

$$(\bar{x}^T, \bar{z}^T)\begin{pmatrix} K & L \\ M & N \end{pmatrix}\begin{pmatrix} \bar{x} \\ \bar{z} \end{pmatrix} = \bar{x}^T K\bar{x} + \bar{x}^T L\bar{z} + \bar{z}^T M\bar{x} + \bar{z}^T N\bar{z} = \bar{x}^T K\bar{x} + 2\bar{x}^T L\bar{z} + \bar{z}^T N\bar{z}$$

. (7.32)

Для нахождения минимума (7.32) возьмем производную по $\bar{x}$ и приравняем ее к нулю: $2K\bar{x} + 2L\bar{z} = 0$, $K\bar{x} + L\bar{z} = 0$, т. е.

$$\hat{\bar{x}} = -K^{-1}L\bar{z}$$

. (7.33)

Отсюда следует очень важный факт: **оптимальная оценка гауссовских параметров по гауссовским наблюдениям линейна (есть линейная функция наблюдений $\bar{z}$).**

Конкретизируем оценку (7.33). Для этого воспользуемся формулой Фробениуса обращения блочных матриц:

$$\begin{pmatrix} A & B \\ C & D \end{pmatrix}^{-1} = \begin{pmatrix} T^{-1} & -T^{-1}BD^{-1} \\ -D^{-1}CT^{-1} & D^{-1} + D^{-1}CT^{-1}BD^{-1} \end{pmatrix},$$

(7.34)

где A и D – квадратные матрицы и $T = A - BD^{-1}C$. Из (7.31) и (7.34) имеем

$T = V_{xx} - V_{xz}V_{zz}^{-1}V_{zx}$, $K = T^{-1}$, $L = -T^{-1}V_{xz}V_{zz}^{-1}$, $-K^{-1}L = TT^{-1}V_{xz}V_{zz}^{-1} = V_{xz}V_{zz}^{-1}$

. Таким образом, оптимальная оценка (7.33) принимает вид

$$\hat{\bar{x}} = V_{xz}V_{zz}^{-1}\bar{z}$$

. (7.35)

Исследуем свойства оптимальной оценки (7.35). Найдем сначала ковариации ошибок этой оценки:

$$M[(\hat{\bar{x}} - \bar{x})(\hat{\bar{x}} - \bar{x})^T] = M[(V_{xz}V_{zz}^{-1}\bar{z} - \bar{x})(\bar{z}^T V_{zz}^{-1}V_{zx} - \bar{x}^T)] =$$

$$= M[V_{xz}V_{zz}^{-1}\bar{z}\bar{z}^T V_{zz}^{-1}V_{zx} - \bar{x}\bar{z}^T V_{zz}^{-1}V_{zx} - V_{xz}V_{zz}^{-1}\bar{z}\bar{x}^T + \bar{x}\bar{x}^T] =$$

$$= V_{xz}V_{zz}^{-1}M[\bar{z}\bar{z}^T]V_{zz}^{-1}V_{zx} - M[\bar{x}\bar{z}^T]V_{zz}^{-1}V_{zx} - V_{xz}V_{zz}^{-1}M[\bar{z}\bar{x}^T] - M[\bar{x}\bar{x}^T]$$

$$= V_{xz} - V_{xz}V_{zz}^{-1}V_{zx},$$

$$M[(\hat{\bar{x}} - \bar{x})(\hat{\bar{x}} - \bar{x})^T] = V_{xx} - V_{xz}V_{zz}^{-1}V_{zx} = T$$

. (7.36)

Итак, матрица T в (7.34) есть матрица ковариаций ошибок оптимальной оценки (7.35). При этом дисперсии ошибок равны диагональным элементам матрицы T.

Найдем ковариации ошибок оценок и наблюдений:

$$M[(\hat{\bar{x}} - \bar{x})\bar{z}^T] = M[(V_{xz}V_{zz}^{-1}\bar{z} - \bar{x})\bar{z}^T] = V_{xz}V_{zz}^{-1}M[\bar{z}\bar{z}^T] - M[\bar{x}\bar{z}^T] = 0$$

.

Это очень важное обстоятельство: **ошибки оптимальных оценок не коррелированы с наблюдениями:**

$$M[(\hat{\bar{x}} - \bar{x})\bar{z}^T] = 0$$

. (7.37)

Отметим, что оценка (7.35) есть **оптимальная линейная оценка $\bar{x}$ для любых центрированных векторов $\bar{x}$ и $\bar{z}$**, т. е. это

оптимальная оценка среди оценок вида $F\bar{z}$, где F – любая матрица. **Но эта оценка не обязательно оптимальна для негауссовских векторов**.

Рассмотрим теперь важный случай оценки одного параметра, т. е. пусть $\bar{x} = x$ и $\hat{x} = \bar{\alpha}^T \bar{z}$, где $\bar{\alpha}$ – весовой вектор оценки. Тогда (7.37) принимает вид

$$M[(\bar{\alpha}^T \bar{z} - x)\bar{z}^T] = 0,$$

где

$$\bar{\alpha}^T = V_{xz} V_{zz}^{-1}, \quad \bar{\alpha} = V_{zz}^{-1} V_{xz}. \tag{7.38}$$

Из (7.38) следует, что $\bar{\alpha}$ является решением системы линейных уравнений

$$V_{zz} \bar{\alpha} = V_{xz}$$

или

$$\begin{pmatrix} V_{z_1 z_1} & V_{z_1 z_2} & \dots & V_{z_1 z_n} \\ \dots & \dots & \dots & \dots \\ V_{z_n z_1} & V_{z_n z_2} & \dots & V_{z_n z_n} \end{pmatrix} \begin{pmatrix} \alpha_1 \\ \dots \\ \alpha_n \end{pmatrix} = \begin{pmatrix} V_{z_1 x} \\ \dots \\ V_{z_n x} \end{pmatrix}. \tag{7.39}$$

При этом средний квадрат ошибки оценки определяется из (7.36):

$$M[(\hat{x} - x)^2] = V_{xx} - V_{xz} V_{zz}^{-1} V_{zx} = \sigma^2 - V_{xz} V_{zz}^{-1} V_{zx}. \tag{7.40}$$

Этот средний квадрат ошибки можно представить в другом виде. Рассмотрим определитель полной ковариационной матрицы V (наблюдений и оцениваемых параметров):

$$|V| = \begin{vmatrix} V_{xx} & V_{xz} \\ V_{zx} & V_{zz} \end{vmatrix}. \tag{7.41}$$

Умножим слева матрицы нижнего ряда на $V_{xz} V_{zz}^{-1}$ и вычтем полученные произведения из матриц верхнего ряда (определитель при этом не изменится):

$$|V| = \begin{vmatrix} V_{xx} - V_{xz} V_{zz}^{-1} V_{zx} & V_{xz} - V_{xz} V_{zz}^{-1} V_{zz} \\ V_{zx} & V_{zz} \end{vmatrix} = |V_{zz}| \, |V_{xx} - V_{xz} V_{zz}^{-1} V_{zx}|.$$

Из (7.40), (7.41) и последнего равенства следует, что

$$M[(\hat{x} - x)^2] = |V| / |V_{zz}|, \tag{7.42}$$

т. е. **средний квадрат ошибки оценки равен отношению определителя полной ковариационной матрицы к определителю ковариационной матрицы наблюдений.**

Пример 1. Рассмотрим оценку гауссовского сигнала *x* с параметрами (0, s2) по зашумленному наблюдению *z=x+q*, где q – гауссовский шум с параметрами (0, s2q), независимый от x. В нашем случае, используя (7.35), получаем

$$V_{zz} = M[z^2] = M[(x+\theta)^2] = \sigma_x^2 + \sigma_\theta^2$$
$$V_{xz} = M[xz] = M[x(x+\theta)] = \sigma_x^2,$$
$$a = V_{xz}V_{zz}^{-1} = \sigma_x^2(\sigma_x^2+\sigma_\theta^2)^{-1} = \frac{\sigma_x^2}{\sigma_x^2+\sigma_\theta^2}.$$

Итак, оптимальной оценкой является

$$\hat{x} = \frac{\sigma_x^2}{\sigma_x^2+\sigma_\theta^2} z \qquad (7.43)$$

Дисперсию ошибки этой оценки получим из (7.36):

$$M[(\hat{x}-x)^2] = V_{xx} - V_{xz}V_{zz}^{-1}V_{zx} = \sigma_x^2 - \sigma_x^2\frac{1}{\sigma_x^2+\sigma_\theta^2}\sigma_x^2 = \sigma_x^2\left(1-\frac{\sigma_x^2}{\sigma_x^2+\sigma_\theta^2}\right) = \frac{\sigma_x^2\sigma_\theta^2}{\sigma_x^2+\sigma_\theta^2},$$

или, по-другому, из (7.42):

$$M[(\hat{x}-x)^2] = \frac{\begin{vmatrix}\sigma_x^2 & \sigma_x^2\\ \sigma_x^2 & \sigma_x^2+\sigma_\theta^2\end{vmatrix}}{\sigma_x^2+\sigma_\theta^2} = \frac{\sigma_x^2\sigma_\theta^2}{\sigma_x^2+\sigma_\theta^2}.$$

Геометрическая интерпретация оптимальной оценки.

Лемма об ортогональном проектировании

Равенству (7.37) можно придать геометрический смысл. Будем считать центрированные СВ **элементами (векторами) векторного пространства**, в котором введено **скалярное произведение** $(xy) = M[xy]$. **Векторы x и y ортогональны, если (xy)=0, т. е. если x и y не коррелированы.** Множество всех линейных комбинаций

вида $\alpha^T z$ есть **<u>линейное пространство</u> Z, натянутое на пространство наблюдений (коротко – пространство наблюдений).** Оптимальную линейную оценку, таким образом, нужно искать в пространстве наблюдений. Оптимальной является оценка $\hat{x}$, при которой значение $M[(\hat{x}-x)^2]$ минимально, т. е.

$$M[(\hat{x}-x)(\hat{x}-x)] = (\hat{x}-x, \hat{x}-x) = \|\hat{x}-x\|^2$$

принимает минимальное значение. Следовательно, длина вектора $\hat{x}-x$ должна быть **минимальной**. Этот минимум достигается, если <u>вектор</u> $\hat{x}-x$ **ортогонален к пространству** Z, а, следовательно, и к каждому из векторов $z_1, z_2, \ldots, z_n$, что, собственно, и означает равенство (7.37). Отсюда получаем следующее утверждение.

Лемма об <u>ортогональном проектировании</u>. Оптимальная линейная оценка $\hat{x}$ параметра x по наблюдениям $\bar{z}$ есть **ортогональная проекция x на пространство наблюдений** Z. Она удовлетворяет равенству (7.37), а в общем случае оценки $\hat{x} = Fz$ векторного параметра $\bar{x}$ – равенству

$$M[(Fz - \bar{x})z^T] = 0.$$

Оптимальная оценка как условное среднее

Рассмотрим снова задачу оценки параметра x по наблюдениям $\bar{z}$ при квадратичной функции потерь, не предполагая гауссовость x и $\bar{z}$. Тогда оптимальная оценка $\hat{x}$ находится из условия минимума

$$M[(\hat{x}-x)^2|z], \qquad (7.44)$$

где используется условное среднее при заданных наблюдениях. При этом $\hat{x} = \hat{x}(z)$. Приравнивая производную от (7.44) по $\hat{x}$ к нулю, получаем

$$M[x - \hat{x}|z] = 0, \quad M[x|z] - M[\hat{x}|z] - M[x|z] - \hat{x} - 0.$$

Для векторного параметра аналогично:

$$\hat{x} = M[x|z]. \qquad (7.45)$$

Таким образом, **оптимальная оценка в смысле минимума среднего квадрата ошибки есть условное математическое ожидание оцениваемого параметра при заданных значениях наблюдений.** Оценка (7.27) есть частный случай оценки (7.45). Действительно,

$$w(x,z) = w(x|z)w(z), \qquad (7.46)$$

поэтому максимум $w(\bar{x},z)$ по $\bar{x}$ достигается в той же точке, что и максимум $w(x|z)$. Следовательно, (7.45) можно заменить на

$$w(\hat{x}|z) = \max w(x|z). \qquad (7.47)$$

Но у гауссовских распределений максимум ПРВ приходится на среднее значение, поэтому оптимальная оценка и есть условное среднее.

7.7. Априорная неопределенность и способы неполного статистического описания

Реализация байесова подхода в идеальном виде требует достаточно полного статистического описания наблюдений Z и скрытых

параметров q, позволяющего однозначно определить распределения p(q) и P(z|q), которые требуются для нахождения ожидаемой величины потерь (апостериорного риска) при решении u.

В действительности столь полное описание имеется не всегда. Чаще всего имеется некоторая априорная неопределенность, т. е**. неполнота описания.** Обычно относительно Z и q имеется какая-то информация, которая не позволяет считать поставленную задачу совсем бессмысленной, но в то же время не дает возможности воспользоваться байесовым подходом в идеальном виде.

Рассмотрим несколько возможных способов неполного статистического описания.

Неполное описание распределения скрытых параметров

Случай А1. Крайний случай, когда относительно q ничего не известно, кроме области $\otimes$ допустимых значений. В этом случае априорное распределение $p(\theta)$ вообще неизвестно – это может быть любая неотрицательная функция с единственным условием $\int p(\theta)d\theta - 1$. В таких условиях и приходится решать СИИ задачу синтеза решающего правила.

Случай А2. О распределении $p(\theta)$ ничего не известно, но компоненты векторного параметра $q = (q_1,\ldots, q_n)$ связаны функциональными ограничениями $F_1(q_1,\ldots, q_n) = 0, \ldots, F_k(q_1,\ldots, q_n) = 0$, следующими из особенностей решаемой задачи. Используя эти ограничения, компоненты q_i можно привести к виду $q_1 = f_1(a), \ldots, q_n = fn(a)$, т. е. $q = q(a)$, где $a = (a_1,\ldots, a_m)$ – векторный параметр, размерность m которого меньше, чем размерность n параметра q. В результате для статистического описания q достаточно задать распределение $p(\theta)$ на пространстве меньшей размерности, что предпочтительнее.

Случай А3. Распределение параметров q неизвестно, но известны некоторые его статистические характеристики, например, математические ожидания, дисперсии, ковариации и т. п. Тогда о $p(\theta)$ известно не только, что $\int p(\theta)d\theta = 1$, но и что $\int f_k(\theta)p(\theta)d\theta = a_k$, k=1,2,…,K, где $f_k(q)$ – некоторые функции. Например, если известна ковариационная матрица $R=(r_{ij})$ параметров q, то (при нулевых средних) $\int \theta_i\theta_j p(\theta)d\theta = r_{ij}$. Подобные данныесужают класс возможных распределений $p(\theta)$, т. е. уменьшают априорную неопределенность.

Случай А4. Заданы распределения вероятностей низшего порядка, например, маргинальные $p_i(q_i)$, i=1,…,n, или условные $p_i(q_i|q_{i-1})$, i=2,…,n.

Отметим, что описание становится полным, если в первом случае компоненты независимы (тогда $p(q_1,\ldots, q_n) = p_1(q_1)\, p_2(q_2)\ldots..p_n(q_n)$), а во втором – марковские (тогда $p(q_1,\ldots, q_n) = p_1(q_1)\, p_2(q_2|q_1)\, p_3(q_3|q_2)\ldots.. p_n(q_n|q_{n-1})$).

Случай А5. Могут быть априорные сведения качественного характера, например, что компоненты q независимы и одинаково распределены (тогда $p(q_1,\ldots, q_n)=p_0(q_1)\ldots..p_0(q_n)$, где $p_0(*)$ – неизвестное распределение).

Случай А6. Известен тип распределения параметров q, например, что они гауссовские, тогда

$$p(\bar{\theta}) = p(\theta_1,\ldots,\theta_n) = \frac{1}{(2\pi)^{n/2}\det{}^{1/2} R}\exp\left(-\frac{1}{2}(\bar{\theta}-\bar{m})^T R^{-1}(\bar{\theta}-\bar{m})\right),$$

где средние значения $\bar{m}$ и ковариационная матрица R неизвестны.

Общей чертой всех рассмотренных примеров является то, что в условиях априорной неопределенности вместо единственного

распределения $p(\theta)$ параметров q можно задать только класс P0 таких распределений.

Таким образом, исходным описанием параметров q в случае априорной неопределенности является задание класса P0 возможных распределений $p(\theta)$ параметров q.

Чем шире класс P0, тем больше априорная неопределенность. В чисто байесовском случае P0 состоит из единственного элемента $p(\theta)$. В другом крайнем случае (А1) P0 – класс всех возможных распределений на Ω. Все остальные случаи – промежуточные между этими двумя крайними.

Неполное описание наблюдений

Описание априорной неопределенности наблюдений Z аналогично описанию априорной неопределенности параметров q. А именно, имеется целый класс P_θ функций правдоподобия p(z|q).

В чисто байесовском случае P_θ состоит из единственного элемента p(z|q). В другом крайнем случае, когда ничего не известно (подобно случаю А1), P_θ состоит из всех неотрицательных функций p(z|q), удовлетворяющих условию $\int p(z\,|\,\theta)dz = 1$ при всех q.

Параметрическая априорная неопределенность

Параметрический способ является довольно общим и удобным для описания априорной неопределенности. Рассмотрим несколько примеров.

Пример С1. Пусть параметр q дискретен и может принимать значения $q=a_i$, i=1,2,…,n, с вероятностями $p(q=a_i)=p_i$. Если распределение q неизвестно вообще, то в качестве неизвестных параметров, описывающих это распределение, можно взять сами вероятности с очевидными ограничениями $p_i=0$ и $Sp_i=1$, т. е. имеется n-1 неизвестных независимых параметров. Если же известно, например, математическое ожидание m_θ, то добавляется еще одно ограничение $\sum a_i p_i = m_\theta$, а количество независимых параметров уменьшается еще на единицу.

Пример С2. Пусть $q=(q_1,\ldots, q_n)$ – последовательность, описывающая, например, случайный процесс, подлежащий фильтрации. При этом известно, что это авторегрессионный процесс, описываемый уравнением $\theta_k = \rho\theta_{k-1} + \sigma\xi_k$, где ξ_k – независимые стандартные гауссовские СВ, а ρ и σ неизвестны. Тогда

$$p(\theta_k \mid \theta_{k-1}) = p(\theta_k \;\; \theta_{k-1}, \rho, \sigma) = \frac{1}{\sqrt{2\pi}\sigma}\exp\left(-\frac{1}{2\sigma^2}(\theta_k - \rho\theta_{k-1})^2\right)$$

и

$$p(\theta \mid \rho, \sigma) = p(\theta_1,\ldots,\theta_n \mid \rho, \sigma) = \frac{1}{(2\pi)^{n/2}\sigma^n}\exp\left(-\frac{1}{2\sigma^2}(\theta_1^2 + \sum_{k=2}^{n}(\theta_k - \rho\theta_{k-1})^2)\right)_{-}$$

распределение, зависящее от двух неизвестных параметров ρ и σ. Аналогичным образом можно записать распределение $q=(q_{ij}$: i=1,2,…,m, j=1,2,…,n), когда q – И размеров $m\times n$, заданное моделью Хабиби с неизвестными значениями ρ, r и σ.

В общем случае параметрическую неопределенность описания параметров q **можно представить в виде**

$$p(\theta) = p(\theta \mid \beta), \qquad (7.48)$$

где β – совокупность неизвестных параметров.

Пример С3. Пусть совокупность наблюдений $z=(z_1,\ldots,z_n)$ представляет последовательность независимых нормальных компонент с неизвестной дисперсией σ^2 и математическими ожиданиями $M[z_i]=m_i(\theta)$, известным образом зависящими от q. Тогда ФП будет

$$P(z \mid \theta) = P(z \mid \theta, \sigma^2) = \frac{1}{(2\pi\sigma^2)^{n/2}}\exp\left(-\frac{1}{2\sigma^2}\sum_{i=1}^{n}(z_i - m_i(\theta))^2\right), \qquad (7.49)$$

которая содержит один неизвестный параметр σ.

В общем случае параметрическую неопределенность описания наблюдений можно представить в виде

$$P(z \mid \theta) = P(z \mid \theta, \alpha), \qquad (2.50)$$

где α – совокупность неизвестных параметров.

Пример С4. Рассмотрим частный случай примера С3. Пусть q может принимать два значения: q=0 и q=1;

$M[z_i \mid \theta = 0] = m_i(0) = 0$ и

$M[z_i \mid \theta = 1] = m_i(1) = as_i, i = 1,\ldots,n$, где s_i – известные величины и *a* – неизвестный коэффициент.

Этот пример можно трактовать как задачу обнаружения сигнала известной формы $s=(s_1,\ldots,s_n)$, но неизвестной интенсивности *a* на фоне

некоррелированного шума неизвестной интенсивности (дисперсии) σ^2. В этом случае (7.49) принимает вид

$$P(z \mid \theta = 0) = P(z \mid \theta = 0, \sigma^2) = \frac{1}{(2\pi\sigma^2)^{n/2}} \exp\left(-\frac{1}{2\sigma^2}\sum_{i=1}^{n} z_i^2\right),$$

$$P(z \mid \theta = 1) = P(z \mid \theta = 1, \sigma^2, a) = \frac{1}{(2\pi\sigma^2)^{n/2}} \exp\left(-\frac{1}{2\sigma^2}\sum_{i=1}^{n} (z_i - a_i)^2\right),$$

причем при q=0 ФП зависит от одного параметра σ^2, а при q=1 – от двух параметров σ^2 и *a*, но общий вид (7.50) сохраняется, если взять $\beta = (\sigma^2, a)$.

Априорная неопределенность задания функции потерь

Функция потерь g(u,q,z) также может в отдельных задачах иметь неполное описание. Это в особенности касается сферы бизнеса, политики и т. п., когда неясно, к каким последствиям может привести то или иное решение.

В рассматриваемых нами задачах обработки И обычно удается подобрать функцию потерь, соответствующюю понятию оптимальности решения задачи. Поэтому мы будем практически всегда считать, что функция потерь определена.

Отметим, тем не менее, что априорная неопределенность задания функции потерь может быть описана стандартным способом – указанием класса G возможных функций потерь, в том числе, параметрического класса таких функций.

8. Синтез решающих правил в условиях априорной неопределенности. Адаптивные алгоритмы

Априорная неопределенность задания модели данных вносит дополнительные трудности при синтезе алгоритмов обработки этих данных. Алгоритмы должны каким-то образом адаптироваться, т. е. приспосабливаться к конкретным обрабатываемым данным. Синтезу адаптивных алгоритмов вообще и адаптивных алгоритмов обработки И, в частности, уделяется большое внимание исследователей. Разработан ряд общих подходов к решению данной проблемы, а также множество алгоритмов для решения конкретных задач. По этой тематике имеется обширная литература.

8.1. Особенности задачи синтеза при априорной неопределенности

Основная особенность задачи синтеза в условиях априорной неопределенности заключается в невозможности **однозначного** определения распределения вероятностей параметров θ и наблюдений z, а, следовательно, и в невозможности однозначного определения величины среднего риска, который является критерием для выбора оптимального решающего правила.

Как уже отмечалось, при априорной неопределенности можно указать только класс P0 возможных распределений p(θ) параметра θ и класс P_θ возможных функций правдоподобия $\Gamma(z \mid \theta)$. Таким образом, возможно только указание класса $P = (P_0, P_\theta)$ возможных пар распределений p=(p(θ),P(z|θ)).

Тогда для каждого $p \in P$ и каждого правила u = u(z) имеется свой средний риск R(u(z)) = R(u(z),p), являющийся функционалом от u(z) и p. Каждой паре $p \in P$ будет соответствовать свое решающее правило u0p(z) = u0(z,p). При этом неизвестно, какая именно пара p

имеет место в действительной ситуации. Выбрать же можно только одно решающее правило.

Задачей синтеза в этих условиях является отыскание такого правила u=u(z), которое в соответствии с некоторым **установленным порядком** обладало бы **наибольшей предпочтительностью** при всех возможных $p \in P$.

Понятия «установленный порядок» и «наибольшая предпочтительность» должны вводиться дополнительно и будут раскрываться в дальнейшем.

8.2. Существенная и несущественная априорная неопределенность

Пример 1. Пусть в примере С4 п.7.7 a=1 известно, но σ^2 неизвестно; p(ϑ=0)=p(θ=1)=0,5; g(0,0)=g(1,1)=0 и g(0,1)=g(1,0)=1, т. е. за неверное решение дается единичный штраф. Тогда при конкретном значении σ^2 оптимальное решающее правило (основанное на ОП) имело бы вид

$$\Lambda(z)=\frac{P(z\mid\theta=1,\sigma^2)}{P(z\mid\theta=0,\sigma^2)}\begin{cases}\geq:\Rightarrow\theta=1,\\ <1\Rightarrow\theta=0.\end{cases} \qquad (8.1)$$

Подставляя в это правило найденные в примере С4 плотности, получаем $\exp\left(\frac{1}{2\sigma^2}\left(\sum z_i^2-\sum(z_i-s_i)^2\right)\right)\gtrless 0$ или $\sum z_i^2 \gtrless \sum(z_i-s_i)^2$ (если «³», то θ=1; если «<», то θ=0).

Оказалось, что, независимо от значения σ^2, решающее правило одно и то же. Поэтому это правило следует признать оптимальным в условиях данного примера. Априорная неопределенность

(неизвестность σ^2) оказалась несущественной в смысле влияния на структуру решающего правила.

Таким образом, могут встречаться ситуации, когда имеющаяся априорная неопределенность никак не сказывается на решающем правиле – оно остается тем же самым, что и при полном описании. Такая априорная неопределенность называется **несущественной**.

Однако при этом может оказаться, что средний риск, характеризующий возможные потери при использовании этого правила, остается неопределенным. В рассмотренном примере 1 очевидно, что при σ^2 = 0 ошибочных решений не будет и средний риск равен нулю. Но этот риск будет возрастать с ростом σ^2 (покажите, что он стремится к 0.5 при σ^2 ® ¥).

Пример 2. Пусть теперь в примере 1 еще и s_i не известны. Тогда при известных σ^2 и s_i оптимальным решающим правилом было бы снова (как и в примере 1)

$$\sum z_i^2 \gtrless \sum(s_i-s_i)^2,$$

которое, хотя и не зависит от σ^2, но существенно зависит от формы сигнала $S=(s_1,\dots s_n)$. Мы фактически не знаем, что надо искать, поэтому и не можем найти, используя традиционный подход к синтезу решающего правила. Постараемся найти какое-нибудь подходящее правило обнаружения. Преобразуем имеющееся нереализуемое правило к виду

$$\sum z_i^2 \gtrless \sum(z_i-s_i)^2=\sum z_i^2-2\sum s_i z_i+\sum s_i^2,$$ т. е. $$\sum s_i z_i \gtrless 0.5\sum s_i^2.$$

Значение $0.5\sum s_i^2$, хотя и неизвестно, но от наблюдений не зависит, обозначим его через Λ_0. Получаем $\sum s_i z_i \gtrless \Lambda_0$, т. е. правило заключается в сравнении взвешенной суммы $\sum s_i z_i$ с порогом Λ_0. При этом весовыми коэффициентами наблюдений z_i должны быть неизвестные значения $(s_1, \dots, s_n)$ сигнала. Где бы их взять? Если сигнал в наблюдениях присутствует, то с некоторым приближением $s_i \approx z_i$ (по крайней мере, при относительно небольшой дисперсии шума σ^2). Отсюда следует правило

$$\sum z_i^2 \gtrless \Lambda_0 .$$

Осталось только подобрать подходящий порог Λ_0. Отметим, что задачу этого примера можно трактовать как задачу различения гипотез H_0 (все $M[z_i] = 0$) и H_1 (не все $M[z_i] = 0$).

Таким образом, априорная неопределенность может быть и **существенной**, т. е. влиять на структуру решающего правила.

Рассмотрим теперь общий случай. Пусть имеющаяся априорная неопределенность позволяет только задать класс P пар распределений $p = (p(\theta), P(z \mid \theta))$. Найдем апостериорный риск

$$R(u \mid \theta, p) = \int g(u, \theta, z) \frac{P(z \mid \theta) p(\theta)}{\int P(z \mid \theta) p(\theta) d\theta} d\theta \qquad (8.2)$$

для каждой такой пары p. Минимум (8.2) по u есть оптимальное решение для фиксированного p.

Если этот минимум достигается для $u = u(z)$, одинакового при всех $p \in P$, то априорная неопределенность несущественна. Само правило с таким свойством называется **равномерно наилучшим** для заданного класса P.

Существование таких правил является редким случаем. Чаще встречаются приближенно равномерные наилучшие правила.

8.3. Подходы к определению понятия оптимальности в условиях априорной неопределенности

Как уже отмечалось, каждой паре распределений p из класса возможных пар P соответствует, вообще говоря, свое оптимальное решающее правило. При этом неизвестно, какая именно пара p имеет место в действительности в момент принятия решения. Это обстоятельство требует разработки дополнения понятия оптимальности применительно к случаю априорной неопределенности, т. е. установления некоторого порядка предпочтения в множестве решающих правил, учитывающего особенности класса P. Рассмотрим некоторые подходы к определению оптимальности решения в этих новых условиях.

Пусть $\Pi(z)$ – множество всех решающих правил. Для каждого $p \in P$ оптимальное, т. е. байесово, решение $u_0(z) = u_0(z, p)$ определяется из условия минимума среднего риска:

$$\min_{u(z)} R(u(z), p) = R(u_0(z, p), p) . \qquad (8.3)$$

Представим эту зависимость решения $u_0(z, p)$ от p в виде условного графика (рис. 8.1), где по оси абсцисс откладывается p, а по оси ординат – соответствующие $u_0(z, p)$. Множество всех $u_0(z, p)$, $p \in P$, составляет некоторое подмножество правил $U_0(z) \subset U(z)$, из которых и следует выбирать окончательное решающее правило.

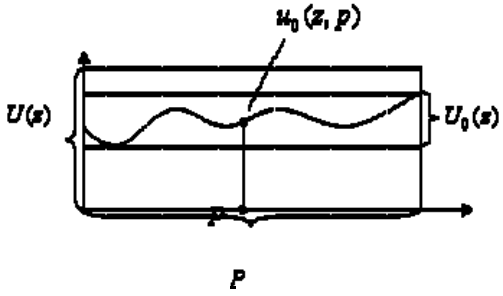


Рис. 8.1.

Равномерно наилучшее решение

Иногда при всех $p \in P$ байесово решение $u_0(z, p)$ – одно и то же, т. е. $u_0(z)$ (рис. 8.2). Тогда это решение является равномерно наилучшим, оно абсолютно оптимально, априорная неопределенность несущественна. Решение $u_0(z)$ может быть найдено с помощью обычного байесова подхода при произвольном p.

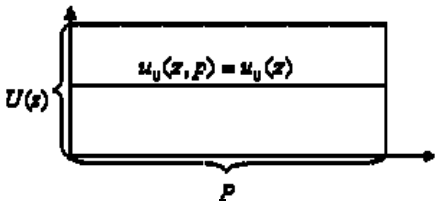


Рис. 8.2.

Отметим, что если ввести произвольную меру $\mu(p)$ на P (не обязательно вероятностную), проинтегрировать средний риск по этой мере:

$$\overline{R(u(z))} = \int_P R(u(z), p) d\mu(p) \tag{8.4}$$

и найти правило $u^*(z)$, минимизирующее этот функционал, то при существовании равномерно наилучшего решения $u_0(z)$ окажется, что $u_0(z) = u^*(z)$. Это означает, что в рассматриваемом случае можно произвольным образом усреднять средний риск. Такое усреднение может существенно упростить поиск решения, так как усредненный риск (8.4) уже не зависит от p.

К сожалению, равномерно наилучшие правила в реальных задачах встречаются редко. Чаще может встретиться ситуация, когда разброс значений среднего риска $R(u(z), p)$ относительно невелик при всех $p \in P$ и $u(z) \subset U_0(z)$. Тогда можно получить приблизительно равномерно наилучшее решение, взяв $u_0(z)$, при котором $R(u(z), p)$ принимает значения, по возможности наиболее близкие к среднему из значений $R(u(z), p)$.

Принцип минимума усредненного риска

Рассмотрим (8.4) при некоторой мере $\mu(p)$ на P и введем принцип предпочтения решений по **минимуму усредненного среднего риска**:

$$\bar{R}(u^*(z)) = \min_{u(z)} \bar{R}(u(z)) = \min_{u(z)} \int_P R(u(z), p)\, d\mu(p)$$

. (8.5)

Неоднозначность выбора решений $u^*(z)$ будет связана только с неоднозначностью выбора меры усреднения $\mu(p)$. Этой мере может быть дана различная трактовка.

Во-первых, $\mu(p)$ может рассматриваться как некоторое априорное распределение вероятностей на P. Например, можно взять равномерное распределение, если есть основания считать p равновозможными. Отметим, что при задании вероятностной меры $\mu(p)$ задача становится чисто байесовской, так как становится возможным определить распределение на z и θ:

$$p(z,\theta) = \int_P p(\theta) P(z \mid \theta) d\mu(p)$$

. (8.6)

Таким образом, в чистом виде этот случай соответствует полной определенности описания z и λ, поэтому неинтересен. Более интересен случай, когда мера $\mu(p)$ известна приближенно.

Во-вторых, $\mu(p)$ можно рассматривать как меру, характеризующую значимость последствий от принятия решений в условиях $p = (p(\theta), P(z \mid \theta))$.

Принципы минимакса (крайнего пессимизма), крайнего и умеренного оптимизма

Если не существует точного или приближенного равномерного наилучшего решения, то одним из возможных принципов предпочтения является **принцип минимакса:** выбирается правило $u(z) = u_M(z)$, при котором

$$\min_{u(z)} \max_{p \in P} R(u(z), p) = \max_{p \in P} R(u_M(z), p)$$

. (8.7)

проиллюстрируем этот принцип рис. 8.3, на котором приведены графики зависимости среднего риска $R(u(z), p)$ от p для различных $u(z) \subseteq U(z)$.

Максимальные значения $R(u(z), p)$ по p для каждого $u_i(z)$ отмечены кружками, минимальные – квадратами, а средние арифметические максимума и минимума – звездочками. Принципу минимакса (8.7) на рис. 8.3 удовлетворяет решение $u_M(z) = u_4(z)$. Это правило при любой ситуации p обеспечивает средние потери, не превышающие $R(u_4(z),p_1)$, где p_1 – точка максимума функции $R(u_4(z),p)$ по p. Любое другое решение при некоторых p может привести к большим потерям, хотя и при других p – к меньшим.

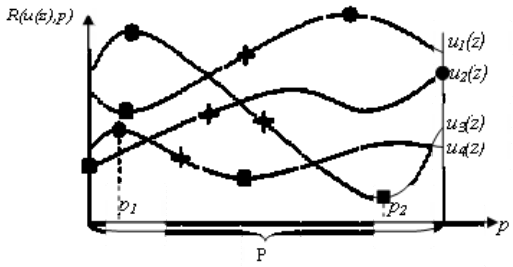


Рис. 8.3.

Таким образом, принцип минимакса обеспечивает минимальные средние потери в наихудшей из возможных ситуаций. Поэтому этот принцип называется **принципом крайнего пессимизма**, поскольку предполагается, что случится самая плохая из возможных ситуаций.

Возможен и другой крайний подход к определению понятия предпочтения решений – **принцип крайнего оптимизма**. А именно, будем предполагать, что случится наиболее благоприятная из возможных ситуаций. Тогда выбирается правило $u(z) = u_m(z)$, при котором

$$\min_{u(z)} \min_{p \in P} R(u(z), p) = \min_{p \in P} R(u_m(z), p) \quad (8.8)$$

На рис. 8.3 этому принципу соответствует правило $u_m(z) = u_3(z)$.

Между этими крайностями располагается **принцип умеренного оптимизма-пессимизма**, исходящий из предпосылки, что обычно случается что-то среднее между наилучшим и наихудшим вариантами. Правило $u(z) = u_s(z)$ выбирается из условия

$$\min_{u(z)} [\alpha \min_{p \in P} R(u(z), p) + (1-\alpha) \max_{p \in P} R(u(z), p)] =$$
$$= \alpha \min_{p \in P} R(u_s(z), p) + (1-\alpha) \max_{p \in P} R(u_s(z), p), \quad (8.9)$$

где α – константа между 0 и 1. При α = 0 правило (8.9) превращается в (8.7), а при α = 1 – в (8.8). При α = 0.5 в (8.9) будут использоваться средние арифметические максимумов и минимумов средних потерь. На рис. 8.3 эти значения отмечены крестиками, и для этого примера будет выбрано $u_s(z) = u_4(z)$.

Принцип асимптотической оптимальности

В практических задачах с априорной неопределенностью часто имеется большой объем наблюдений, которые в случае полного описания могут быть даже избыточны для решения данной задачи. Но

в условиях априорной неопределенности эти избыточные данные могут оказаться полезными.

Пример 3. Рассмотрим задачу двухальтернативного решения при u = 0 или u = 1; $\theta = 0$ или $\theta = 1$; $P(\theta=0) = P(\theta=1) = 0.5$; g(0,0) = g(1,1) = 0, g(0,1) = g(1,0) = 1; наблюдение $z = (z_1, z_2, \ldots z_n, z_{n+1})$ есть совокупность независимых СВ с ПРВ

$$P(z \mid \theta = 1) = P_1(z_{n+1}) \prod_{i=1}^{n} P_0(z_i \mid \alpha),$$
$$P(z \mid \theta = 0) = \prod_{i=1}^{n+1} P_0(z_i \mid \alpha), \quad (8.10)$$

где α – параметр ПРВ величин $z_1, \ldots, z_n$. Таким образом, $P_1(t)$ известна полностью, а $P_0(t/\alpha)$ – с точностью до параметра α. Неизвестность α и составляет априорную неопределенность. Этот пример можно трактовать как **задачу обнаружения сигнала** (θ=1 и u=1), который может проявиться только в последнем наблюдении z_{n+1}. Если сигнал есть, то z_{n+1} имеет ПРВ $P_1(z_{n+1})$, если же его нет, то ПРВ z_{n+1} (как и всех остальных наблюдений) есть $P_0(z_i|\alpha)$.

Если бы параметр α был известен (априорной неопределенности нет), то оптимальное решающее правило имело бы вид

$$\Lambda(z) = \frac{P(z \mid \theta = 1)}{P(z \mid \theta = 0)} = \frac{P_1(z_{n+1})}{P_0(z_{n+1} \mid \alpha)} \begin{cases} \geq 1 \Rightarrow u = 1, \\ < 1 \Rightarrow u = 0. \end{cases} \quad (3.11)$$

Это правило зависит только от последнего наблюдения z_{n+1}, а все остальные данные $z_1, \ldots, z_n$ избыточны, их могло бы и не быть вообще.

Если же α неизвестно, то данные $z_1, \ldots, z_n$ можно попытаться использовать для уменьшения априорной неопределенности. Более того, возможно, что с ростом количества этих данных влияние априорной неопределенности будет уменьшаться (по крайней мере, не

возрастать). Можно даже надеяться, что при увеличении количества и качества данных удастся получить столь же хорошее решение, как и при отсутствии априорной неопределенности.

Отсюда вытекает **принцип асимптотической оптимальности**: наиболее предпочтительным является такое правило u = u(z), для которого средний риск R(u(z),p) с увеличением объема данных z стремится к минимальному байесову риску $\min_{u(z)} R(u(z), p)$ для всех $p \in P$ равномерно.

Однако может оказаться, что этот принцип не определяет решения однозначно, так как может быть целый ряд асимптотически оптимальных решений.

8.4. Адаптивный байесов подход

Сущность этого подхода состоит в следующем. Пусть имеется существенная априорная неопределенность в описании параметров θ и/или наблюдений z. Эта неопределенность не позволяет применить обычный байесов формализм: **найти для каждого правила u(z) величину среднего риска R(u(z)), величину апостериорного риска R(u,z) и определить положение минимума R(u,z) по u для каждого z, что и дает оптимальное байесово решение u = u_0(z).**

Отметим, что невозможность применения этого формализма связана **именно с незнанием, а не с существованием**: на самом деле в любых конкретных условиях существуют вполне определенные (хотя и неизвестные) истинные значения R(u(z))=$R_{ист}$(u(z)) для всех u(z), а следовательно, существует и оптимальное решение u_0(z), обращающее $R_{ист}$(u(z)) в минимум.

Попытаемся применить этот байесов формализм в условиях априорной неопределенности, используя сведения, содержащиеся в наблюдениях z для оценки истинного значения апостериорного риска, чтобы получить его приближенное значение $\hat{R}(u,z)$. После нахождения

оценки $\hat{R}(u,z)$ остается воспользоваться стандартным байесовым подходом, используя $\hat{R}(u,z)$ вместо $R_{ист}$(u,z).

Таким образом, адаптивный байесов подход в своей основе ничем не отличается от неадаптивного: главным остается минимизация среднего (апостериорного) риска. Различие только в способе достижения этой цели. При наличии априорной неопределенности приходится модифицировать обычный байесов формализм, вводя в него дополнительные процедуры. При этом повышается роль наблюдений z – они уже не просто аргументы известной функции R(u,z), но еще используются для ее восстановления.

Процесс восстановления функции R(u,z) называется адаптацией, а полученные таким способом правила называются адаптивными байесовыми решающими правилами.

Итак, адаптивный байесов подход основан на замене точной меры ожидаемых потерь ее оценкой на основе имеющихся данных. В этом аспекте различные способы адаптации есть различные способы оценки меры потерь.

Продолжение примера 3. Правило (8.11) нельзя применить, так как неизвестно значение α. Используем наблюдения $z_1,\ldots,z_n$ для нахождения оценки $\alpha^* = \alpha^*(z_1,\ldots,z_n)$. Можно, в частности, взять оценку ММП α *, определяемую в нашем случае соотношением

$$\max_{\alpha} P(z \mid \alpha) = \max_{\alpha} \prod_{i=1}^{n} P_0(z_i \mid \alpha) = \prod_{i=1}^{n} P_0(z_i \mid \alpha^*) .$$

Подставляя это значение α * в (8.11), получим **адаптивное правило**

$$\frac{P_1(z_{n+1})}{P_0(z_{n+1} \mid \alpha^*)} \gtrless 1 . \qquad (8.12)$$

Если ошибка $|\alpha^* - \alpha|$ мала, то можно ожидать, что правило (8.12) будет для подавляющего большинства значений z_{n+1} давать то же решение, что и (8.11).

Для иллюстрации конкретизируем этот пример, взяв $\alpha > 0$ и нормальные распределения

$$P_0(z_1 \mid \alpha) = \frac{1}{\sqrt{2\pi}\sigma} \exp(-(z_1 - \alpha)^2 / 2\sigma^2)$$

и

$$P_1(z_{n+1}) = \frac{1}{\sqrt{2\pi}\sigma} \exp(-z_{n+1}^2 / 2\sigma^2);$$

Тогда (8.11) приводится к виду

$$u(z) = \begin{cases} 1, z_{n+1} \le \alpha/2, \\ 0, z_{n+1} > \alpha/2, \end{cases} \qquad (8.13)$$

а правило (8.12) – к виду

$$u^*(z) = \begin{cases} 1, z_{n+1} \le \alpha^*/2, \\ 0, z_{n+1} > \alpha^*/2. \end{cases} \qquad (8.14)$$

Оценкой ММП параметра α в нашем случае будет $\alpha^* = \sum z_i / n$. При этом ошибка $e = \alpha^* - \alpha$ нормальна с нулевым средним и дисперсией s_2/n. Таким образом, вместо области $A = (-\infty, \alpha/2]$ принятия решения u=1 в правиле (8.13) имеем область $A^* = (-\infty, \alpha^*/2] = (-\infty, \alpha/2 + e/2]$ в правиле (8.14). Эти области различаются на интервал ($\alpha/2$; $\alpha/2+e/2$] или ($\alpha/2+e/2$; $\alpha/2$]. При этом

e/2 имеет СКО $\sigma/2\sqrt{n}$, стремящееся к нулю при $n \to \infty$, поэтому разница между правилами тоже сходит на нет.

Найдем средний риск для правил (8.13) и (8.14). Средний риск для (8.13):

$$R^0 = M[g(u(z),\theta)] = 0.5P(z_{n+1} \le \alpha/2 \mid \theta = 0) + 0.5P(z_{n+1} > \alpha/2 \mid \theta = 1) =$$
$$= 0.5\Phi((\alpha/2 - \alpha)/\sigma) + 0.5(1 - \Phi((\alpha/2 - 0)/\sigma)) = 1 - \Phi(\alpha/2\sigma).$$

Средний риск для (8.14):

$$R^* = M[g(u^*(z),\theta)] = 0.5P(z_{n+1} \le \sum_{i=1}^{n} z_i/2n \mid \theta = 0) + 0.5P(z_{n+1} > \sum_{i=1}^{n} z_i/2n \mid \theta = 1) =$$
$$= 0.5P(z_{n+1} - \sum_{i=1}^{n} z_i/2n \le 0 \mid \theta = 0) + 0.5P(z_{n+1} - \sum_{i=1}^{n} z_i/2n > 0 \mid \theta = 1).$$

Величина $z_{n+1} - \sum_{i=1}^{n} z_i/2n$ нормальна. При θ=0 она имеет среднее значение $\alpha/2$ и дисперсию $(1+1/4n)s_2$, а при θ=1 имеет среднее - $\alpha/2$ и ту же дисперсию. Поэтому

$$R^* = 0.5\Phi((0 - \alpha/2)/\sigma\sqrt{1+1/4n}) + 0.5(1 - \Phi((0 + \alpha/2)/\sigma\sqrt{1+1/4n})) = 1 - \Phi(\alpha/2\sigma\sqrt{1+1/4n})$$

.

Естественно, что $R^* > R^0$, но разница $R^* - R^0 = \Phi(\alpha/2\sigma) - \Phi(\alpha/2\sigma\sqrt{1+1/4n})$ стремится к нулю при $n \to \infty$ и мала уже при относительно небольших n. Таким образом, адаптивное правило (8.14) **асимптотически оптимально**: при увеличении объема данных средний риск R*стремится к минимальному байесову риску R^0, который мы имели бы при отсутствии априорной неопределенности, т. е. при известном α.

Пример 4. Имеется И $Z = \{z_{ij}\}$. На нем, возможно, есть области, отличающиеся от своего окружения большей средней яркостью. **Требуется обнаружить эти области**. Такая задача возникает, в частности, при поиске акваторий, перспективных для рыбного промысла – таковыми являются участки моря с более высокой средней температурой, для такого поиска используются **тепловизионные И моря**.

Зададимся формой области W и ее окружением R, например, квадрат и рамка, показанные на рис. 8.4. Если m_W и m_R – математические ожидания отсчетов изображения по W и по R, то нам нужно выбрать одну из двух альтернатив $H_0(m_W \le m_R)$ или $H_1(m_W > m_R)$. При известных m_W и m_R решающее правило элементарно:

$$m_W - m_R \begin{cases} \le 0 \Rightarrow H_0, \\ > 0 \Rightarrow H_1. \end{cases} \qquad (8.15)$$

Это правило нужно применить ко всевозможным положениям W на Z. Однако m_W и m_R неизвестны. Построим адаптивное правило. Для этого применим в качестве оценок неизвестных m_W и m_R средние арифметические наблюдений:

$$\bar{m}_W = \frac{1}{n_W}\sum_W z_{ij}, \qquad \bar{m}_R = \frac{1}{n_R}\sum_R z_{ij},$$

Рис. 8.4.

где n_W и n_R – количество элементов в W и R. Эти оценки состоятельны. Но теперь $\bar{m}_W > \bar{m}_R$ не обязательно влечет за собой $m_W > m_R$. Поэтому гипотезу H_1 следует принимать только при $m_W - m_R > m$. Более того, порог m может быть различным для разных положений W, так как изображение Z может быть неоднородным.

Поэтому отнормируем разность $\bar{m}_W - \bar{m}_R$ оценкой СКО в рамке R, т. е. величиной $\bar{\sigma}_R = \sqrt{\frac{n_R}{n_R - 1}\sum_R (z_{ij} - \bar{m}_R)^2}$. В результате получим адаптивное правило

$$\frac{\bar{m}_W - \bar{m}_R}{\bar{\sigma}_R} \begin{cases} \le \Lambda_0 \Rightarrow H_0, \\ > \Lambda_0 \Rightarrow H_1. \end{cases} \qquad (8.16)$$

Если все отсчеты изображения имеют ограниченные дисперсии и постоянные средние в W и R, то правило (8.16) сходится к правилу (8.15), когда n_W и n_R стремятся к бесконечности. При этом порог L_0 стремится к нулю.

8.5. Классификация адаптивных алгоритмов

Как уже отмечалось, **одним из способов решения задач обработки данных в условиях априорной неопределенности является применение адаптивных алгоритмов (решающих правил).** В связи с этим отметим основные классы адаптивных алгоритмов.

Аргументные и критериальные задачи

По цели обработки данных адаптивные алгоритмы можно разделить на **аргументные и критериальные**. Исходной посылкой для **синтеза алгоритмов является минимизация средних потерь, формально выражающихся функционалом качества** $R(\bar{a}, z) = J(a)$**, т. е.**

критерием, который нужно минимизировать по некоторым параметрам $\bar{\alpha}$. Однако требования к этой минимизации могут быть различными.

В **аргументных задачах** целью является возможно более точное отыскание точки минимума $\bar{\alpha}^*$ (возможно, переменной). **К этому типу относятся задачи измерения параметров, фильтрации, прогноза и т. д.** Сам критерий $J(\bar{\alpha})$ может вводиться искусственно и играет роль **меры рассогласования между оценкой и точным значением параметра**. При этом алгоритм обработки часто оказывается одинаковым для широкого класса функций потерь.

В **критериальных задачах** целью является приближение $J(\bar{\alpha})$ к его минимальному значению $J^* = J(\bar{\alpha}^*)$, а сами параметры $\bar{\alpha}$ интереса не представляют и, вообще говоря, могут значительно отличаться от $\bar{\alpha}^*$.

Пусть, например, $\bar{\alpha}$ – весовой вектор линейной оценки $\hat{x} = \bar{\alpha}^T \bar{Z}$ гауссовского параметра x по гауссовским наблюдениям $\bar{Z}$ и $J(\bar{\alpha})$ – средний квадрат ошибки этой оценки. Тогда **поверхности $J(\bar{\alpha}) = c$ являются эллипсоидами с центром в $\bar{\alpha}^*$** (рис. 8.5).

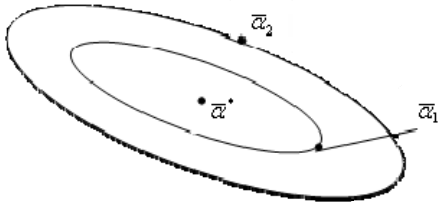


Рис. 8.5.

Может оказаться, что $J(\bar{\alpha}_1) < J(\bar{\alpha}_2)$, хотя и $\bar{\alpha}_1$ находится от $\bar{\alpha}^*$ дальше, чем $\bar{\alpha}_2$. Другими словами, «потребительские» качества вектора $\bar{\alpha}_1$ выше, чем у $\bar{\alpha}_2$, несмотря на то, что $\bar{\alpha}_1$ находится дальше от оптимума, чем $\bar{\alpha}_2$.

Идентификационная и безыдентификационная адаптация

По методу нахождения оптимальных параметров $\bar{\alpha}^*$ адаптивные алгоритмы можно разбить на **идентификационные и безыдентификационные.**

В **идентификационных алгоритмах** сначала по всем имеющимся данным оцениваются все недостающие неизвестные характеристики γ. Затем полученные оценки $\hat{\gamma}$ используются как точные. В результате получаем параметры для алгоритма в виде $\bar{\alpha} = \bar{\alpha}(\hat{\gamma})$. В этом суть многочисленных **модифицированных байесовых решающих правил**. Таким способом было получено решающее правило в примере 3 из п. 8.4.

При всех своих положительных качествах идентификационные алгоритмы имеют следующие серьезные недостатки, особенно при обработке многомерных данных.

1) Зависимость данных от γ может быть очень сложной и даже неизвестной (например, зависимость И от межкадровых смещений), поэтому даже при известных γ определение $\bar{\alpha}(\gamma)$ представляет сложную задачу.

2) Получение оценок $\hat{\gamma}$ требует дополнительных вычислений и делает обработку двухэтапной: сначала находятся оценки, а потом производится собственно обработка, что требует дополнительных линий задержки данных.

3) Дальнейшие вычисления, в которых используются оценки $\hat{\gamma}$, могут быть неустойчивы к ошибкам этих оценок, например, матрицы выборочных корреляций зачастую плохо обусловлены (их детерминанты близки к нулю).

4) Даже точное значение $\bar{\alpha} = \bar{\alpha}(\gamma)$ может отличаться от оптимальных значений параметров алгоритма, так как обрабатываемые данные могут отличаться от используемой для их описания модели.

В алгоритмах **без идентификации** минимизация критерия $J(\bar{\alpha})$ производится по регулируемым параметрам $\bar{\alpha}$ без промежуточных оценок каких-либо характеристик γ исходных данных. При этом $\bar{\alpha}$ могут подбираться итерационно в процессе текущей обработки по наблюдениям за текущими значениями $J(\bar{\alpha})$. Для иллюстрации можно привести пример ручной настройки телевизора во время просмотра телепрограммы.

Для реализации таких алгоритмов необходима оценка текущего значения $J(\bar{\alpha})$, т. е. критерий должен быть наблюдаемым, что является ограничением на область применения этого подхода. Иногда выход может быть найден с помощью замены $J(\bar{\alpha})$ на другой, наблюдаемый критерий $J_1(\bar{\alpha})$, от которого требуется только, чтобы точки минимума $J(\bar{\alpha})$ и $J_1(\bar{\alpha})$ совпадали в аргументных задачах, а в критериальных – чтобы $J(\bar{\alpha})$ приближалось к $J^* = J(\bar{\alpha}^*)$, когда $J_1(\bar{\alpha})$ приближается к $J_1^* = J_1(\bar{\alpha}_1^*)$. Эта методика не означает отхода от адаптивного байесова принципа, поскольку $J(\bar{\alpha})$ по-прежнему минимизируется.

Отметим, что между этими двумя классами алгоритмов есть много общего. В безыдентификационном алгоритме можно найти признаки идентификации. Действительно, поскольку имеется зависимость

$\bar{\alpha} = \bar{\alpha}(\gamma)$, то по найденному $\bar{\alpha}$ можно иногда оценить и γ. Тем не менее, это существенно разные классы алгоритмов.

Квазиоптимальные алгоритмы

Аппроксимация решающего правила

Даже при полном описании данных не всегда удается найти оптимальное решающее правило из-за математических трудностей. Если его и удается найти, оно часто оказывается недопустимо трудоемким. Кроме того, используемая при синтезе модель исходных данных обычно лишь приближенно описывает реальность. В силу этих причин в реальных ситуациях часто не удается найти и применить оптимальное правило.

Поэтому приходится применять **квазиоптимальные**, реализуемые правила, по возможности с меньшим проигрышем в качестве обработки. Для поиска таких правил можно использовать упрощенные модели данных, описывающие лишь их принципиальные свойства. Полученные правила (алгоритмы) содержат некоторые неопределенные параметры $\bar{\alpha}$, которые необходимо выбрать так, чтобы этот алгоритм давал наилучший результат на конкретных обрабатываемых данных.

Итак, к реальным данным адаптируется готовый алгоритм с неизменной структурой, изменяться могут только подстраиваемые параметры $\bar{\alpha}$. Такой подход к адаптации называется **аппроксимацией решающего правила**. Этот прием применяется, например, когда какая-то готовая аппаратура используется для обработки другого класса данных. Другой пример – поиск для решения данной задачи оптимального алгоритма в классе линейных алгоритмов.

8.6. Псевдоградиентные алгоритмы адаптации

Из п. 8.5 можно сделать вывод, что для обработки больших объемов данных (в частности, И и их последовательностей) целесообразно использовать безыдентификационные алгоритмы, учитывая требования простоты и работоспособности при значительных

вариациях реальной ситуации. В значительной степени этим требованиям удовлетворяют **псевдоградиентные** (ПГ) адаптивные алгоритмы.

8.6.1. Структура и общие свойства

Пусть структура процедуры обработки определена, а критерий качества решения задачи сформулирован в терминах минимизации функционала $J(\bar{\alpha})$, который прямо или косвенно отражает средние потери, когда обработка выполняется с параметрами $\bar{\alpha}$. Ввиду априорной неопределенности описания данных нет возможности заранее определить оптимальные параметры $\bar{\alpha}^*$. Поэтому необходима некоторая процедура адаптации, составляющая вместе с процедурой обработки адаптивный алгоритм, в котором параметры $\bar{\alpha}$ определяются на основании конкретной реализации (наблюдения) Z объекта обработки.

Таким образом, задача адаптации формулируется в виде задачи минимизации функции $J(\bar{\alpha}) = J(\bar{\alpha}, Z)$ для конкретной реализации Z, и речь идет об аппроксимации решающего правила, т. е. выбранной процедуры обработки. Применим для решения этой задачи безыдентификационную адаптацию.

Существует ряд численных методов поиска экстремумов. Наиболее распространенными являются различные модификации градиентного алгоритма

$$\bar{\alpha}_n = \bar{\alpha}_{n-1} - \mu_n \nabla J(\bar{\alpha}_{n-1}), \qquad (8.17)$$

где $\bar{\alpha}_n$ – следующее за $\bar{\alpha}_{n-1}$ приближение к точке минимума; μ_n – положительная числовая последовательность, определяющая длину шагов; $\nabla J(\bar{\alpha})$ – градиент функции $J(\bar{\alpha})$. Каждый шаг в (8.17)

делается в направлении скорейшего убывания $J(\bar{\alpha})$. Хотя и при выполнении некоторых условий сходимость $\bar{\alpha}_n \to \bar{\alpha}^*$ имеет место, она может оказаться очень медленной. Для ее ускорения выбираются направления, отличные от антиградиента (методы Ньютона, сопряженных градиентов и т. д.).

Применению этих методов в обработке И препятствует необходимость многократных и громоздких вычислений $\nabla J(\bar{\alpha}_{n-1}, Z)$, каждое из которых обычно включает в себя всю процедуру обработки Z при параметрах $\bar{\alpha}_{n-1}$. Значительно сократить объем вычислений можно, если вместо $\nabla J(\bar{\alpha}_{n-1}, Z)$ взять усечение $\nabla Q(\bar{\alpha}_{n-1}) = \nabla J(\bar{\alpha}_{n-1}, Z_n)$, т. е. вычислять градиент не по всей реализации Z, а только по некоторой ее части Z_n, например, в скользящем окне на И. Но тогда в (8.17) вместо точного значения градиента будет использоваться его значение со случайной ошибкой $\bar{\delta}_n$, и получается алгоритм

$$\bar{\alpha}_n = \bar{\alpha}_{n-1} - \mu_n(\nabla J(\bar{\alpha}_{n-1}, Z) + \bar{\gamma}_n) = \bar{\alpha}_{n-1} - \mu_n \nabla Q(\bar{\alpha}_{n-1}) \qquad (3.18)$$

.

Последовательность $\bar{\alpha}_n$ становится случайной, поэтому случаен и сам факт ее сходимости к $\bar{\alpha}^*$.

Случайные ошибки $\bar{\delta}_n$ в (8.18), вообще говоря, не являются серьезным препятствием для сходимости $\bar{\alpha}_n \to \bar{\alpha}^*$. Существует большой класс **методов стохастической аппроксимации**, основанных на том факте, что при центрированности ошибки ($M[\bar{\delta}_n] = 0$) процедура (8.18) сходится к $\bar{\alpha}^*$, как и процедура (8.17). Применяются

и так называемые **методы случайного поиска**, в которых ошибка $\bar{\delta}_n$ вводится искусственно. Интересно, что наличие случайной ошибки $\bar{\delta}_n$ может даже ускорять сходимость процедуры.

Смысл центрированности $\bar{\delta}_n$ состоит в том, что шаги процедуры (5.2) **в среднем выполняются точно по антиградиенту**, так сказать «в среднем в правильном направлении». Оказывается, что это условие можно существенно ослабить. Рассмотрим пример, показанный на рис. 8.6, для которого $J(\bar{\alpha})$ есть расстояние между точкой $\bar{\alpha}$ и фиксированной точкой $\bar{\alpha}^*$. Тогда антиградиент $-\nabla J(\bar{\alpha})$ направлен по прямой от $\bar{\alpha}$ к $\bar{\alpha}^*$. Если вместо этого направления все время двигаться под углом, например, 89° к нему, то движущаяся точка $\bar{\alpha}$ по спирали будет стремиться к $\bar{\alpha}^*$. Таким образом, для сходимости (8.18) к $\bar{\alpha}^*$ совсем не обязательно, чтобы $M[\bar{\delta}_n]=0$. Однако, если в нашем примере взять угол 91°, то точка $\bar{\alpha}$ будет удаляться от $\bar{\alpha}^*$ по спирали.

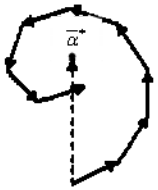


Рис. 8.6

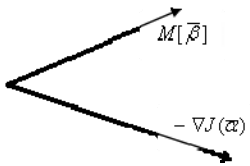


Рис.8.7

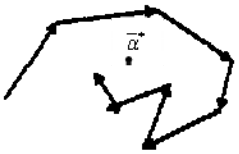


Рис. 8.8

В 1973 г. Я. З. Цыпкиным и Б. Т. Поляком было введено понятие **псевдоградиента** (ПГ), на основе которого разработан единый подход к анализу и синтезу алгоритмов стохастической минимизации функционалов. Класс ПГ алгоритмов очень широк и включает в себя все (или почти все) алгоритмы адаптации и обучения. Эти алгоритмы основаны на процедуре

$$\bar{\alpha}_n = \bar{\alpha}_{n-1} - \mu_n \bar{\beta}_n, \qquad (8.19)$$

где $\bar{\beta}_n$ – случайное (в частности, детерминированное) направление, вообще говоря, зависящее от предыдущих значений $\bar{\alpha}_i$ и от номера шага n. Направление $\bar{\beta}_n$ называется **псевдоградиентом** функционала $J(\bar{\alpha})$ в точке $\bar{\alpha}_{n-1}$, если выполнено **условие псевдоградиентности**

$$[J(\bar{\alpha}_{n-1})]^T M[\bar{\beta}_n] \geq 0, \qquad (8.20)$$

где левая часть есть скалярное произведение, поэтому **ПГ в среднем составляет острый угол с точным значением градиента** (рис. 8.7).

Алгоритм (8.19) называется **псевдоградиентным,** если $\bar{\beta}_n$ является ПГ на каждом шаге. В этом случае шаги в (8.19) будут производиться в среднем в сторону уменьшения $J(\bar{\alpha})$ и можно надеяться на сходимость $\bar{\alpha}_n \to \bar{\alpha}^*$ при $n \to \infty$ (рис. 8.8), хотя и некоторые шаги могут быть сделаны в сторону увеличения $J(\bar{\alpha})$. И действительно, выполнение относительно слабых условий оказывается достаточным для сходимости с вероятностью единица при любом начальном приближении $\bar{\alpha}_0$.

Главным из этих условий является строгое неравенство в (8.20) при $\bar{\alpha}_{n-1} \neq \bar{\alpha}^*$, единственность точки минимума, а также $\sum \mu_n = \infty$ (обеспечивает возможность дойти из любой точки до $\bar{\alpha}^*$) и $\sum \mu_n^2 < \infty$ (обеспечивает возможность асимптотического уменьшения дисперсии колебаний последовательности $\bar{\alpha}_n$). Последним двум условиям удовлетворяют, например, последовательности вида $\mu_n = 1/(a+bn)$.

Как уже отмечалось, не всегда целью является приближение $\bar{\alpha}_n$ к $\bar{\alpha}^*$. В критериальных задачах требуется, чтобы $J(\bar{\alpha}_n) \to J(\bar{\alpha}^*)$. Условия такой сходимости даже слабее условий аргументной сходимости.

Скорость сходимости для обоих классов задач имеет обычный для статистических алгоритмов порядок $O(1/\sqrt{n})$, хотя для критериальных задач она иногда выше, чем для аргументных.

Отметим, что алгоритм (8.19) является существенно более общим, чем (8.18), так как в (8.19) не предполагается возможность вычисления $J(\bar{\alpha})$ или $\nabla J(\bar{\alpha})$, хотя бы и со случайной ошибкой, т. е. $J(\bar{\alpha})$ может быть и **ненаблюдаемым**. Необходимо только наличие **наблюдаемого** ПГ. В частности, в качестве $\bar{\beta}_n$ может быть выбрано (даже зашумленное) значение градиента другого функционала $J_1(\bar{\alpha})$, у которого та же точка минимума, что и у $J(\bar{\alpha})$.

Допустимость зависимости $\bar{\beta}_n$ от предыдущих значений $\bar{\alpha}_i$ дает **возможность применения ПГ алгоритмов для обработки не только**

одномерных данных, но и многомерных в порядке некоторой их развертки.

До сих пор предполагалось, что задачей является нахождение точки минимума $\bar{\alpha}^*$ функционала $J(\bar{\alpha}, Z)$, единой для всей реализации Z. Такая точка $\bar{\alpha}^*$ существует, но обработка будет оптимальной, если данные Z **однородны**. Для сходимости $\bar{\alpha}_n \to \bar{\alpha}^*$ при этом требуется сходимость $\mu_n \to 0$. Если же, начиная с некоторого момента, ограничить μ_n снизу (например, взять постоянные $\mu_n = \mu$), то дисперсии ошибок оценок $\bar{\alpha}_n$ параметров $\bar{\alpha}^*$ перестанут уменьшаться и будут иметь порядок μ^2, а сами $\bar{\alpha}_n$ будут колебаться около $\bar{\alpha}^*$.

Таким образом, если обработку однородных данных производить одновременно с оценкой $\bar{\alpha}^*$ (и при $\mu_n = \mu$), то по достижении установившегося режима будет осуществляться некоторая квазиоптимальная обработка.

Если произойдет скачкообразное изменение характеристик Z или переход к обработке других данных, то могут измениться и значения требуемых оптимальных параметров $\bar{\alpha}_n^*$. Если обработка будет просто продолжена, то непосредственно после этого скачка возможно значительное ухудшение качества обработки, после чего постепенно снова будут достигнуты квазиоптимальные результаты.

При плавном изменении характеристик наблюдений Z (точнее, при плавном изменении оптимальных значений параметров $\bar{\alpha}^*$), соизмеримом со скоростью переходного процесса процедуры (8.19), появляется возможность применения ПГ алгоритмов к обработке неоднородных данных без их сегментации на участки относительно однородной структуры.

В такой постановке алгоритм (8.19) используется для текущей оценки переменных параметров $\bar{\alpha}_n^*$. Вопрос об асимптотической сходимости снимается. Вместо этого требуется как можно более точное приближение $\bar{\alpha}_n$ к изменяющемуся $\bar{\alpha}_n^*$ (или же $J(\bar{\alpha}_n)$ к $J(\bar{\alpha}_n^*)$).

При этом возникает проблема компромисса в выборе μ_n: для уменьшения дисперсии ошибки нужно уменьшить μ_n, а для избежания запаздывания оценки следует увеличить μ_n.

Итак, ПГ алгоритмы просты в реализации, применимы к очень широкому классу однородных и неоднородных данных (причём в случае однородных данных сходятся к оптимальным алгоритмам). Адаптация может выполняться непосредственно в процессе обработки, поэтому не требуется линий задержки данных. Отмеченные положительные качества ПГ адаптивных алгоритмов делают их привлекательными для применения в обработке И, а также других больших массивов данных.

8.6.2. Выбор псевдоградиента

Главным в синтезе ПГ алгоритмов вида (8.19) является нахождение ПГ $\bar{\beta}_n$ функционала качества $J(\bar{\alpha},Z)$. Рассмотрим два важнейших случая.

Случай 1. В большинстве статистических задач (в частности, в обработке И) функционал качества выражается через среднее значение некоторой функции $g(\bar{\alpha},Z)$:

$$J(\bar{\alpha},Z)=M[g(\bar{\alpha},Z)], \qquad (8.21)$$

например, через средний квадрат ошибки оценки параметра θ. Тогда

$$g(\bar{\alpha},Z)=[f(\bar{\alpha},Z)-\theta]^2=\Delta^2(\bar{\alpha},Z), \qquad (8.22)$$

где θ – точное значение параметра и $f(\bar{\alpha},Z)=\theta$ – его оценка. К (8.22) приводят задачи прогноза, фильтрации и т. д.

Если реализации $g(\bar{\alpha},Z)$ наблюдаемы (например, при прогнозе), то можно взять в качестве ПГ $\bar{\beta}_n=\nabla g(\bar{\alpha}_{n-1},Z)$ или его сужение

$$\bar{\beta}_n=\nabla g(\bar{\alpha}_{n-1},Z_n) \qquad (8.23)$$

$$(\nabla M[g])^T M[\bar{\beta}]=(\nabla M[g])^T M[\nabla g]=(M[\nabla g])^T M[\nabla g]=|M[\nabla g]|^2>0.$$

на часть данных Z_n (например, на скользящее окно на И). Если возможно дифференцирование под знаком математического ожидания, то для направления (8.23) условие псевдоградиентности (8.20) выполняется тривиально:

Если же реализации функции (8.22) не наблюдаемы, то следует ввести вспомогательный наблюдаемый функционал качества J_1, выраженный через среднее значение некоторой функции.

Например, при оценке математического ожидания α случайной величины Z можно взять $J_1(\alpha,Z)=M[(Z-\alpha)^2]$, тогда

$$\bar{\beta}_n=-(z_n-\alpha_{n-1}) \quad \text{и} \quad \alpha_n=\alpha_{n-1}+\mu_n(z_n-\alpha_{n-1}), \qquad (8.24)$$

где z_n – очередное наблюдение Z.

При оценке среднего квадрата случайной величины Z можно взять $J_1(\alpha, Z) = M[(Z^2 - \alpha)^2]$, тогда

$$\bar{\beta}_n = -(z_n^2 - \alpha_{n-1}) \quad \text{и} \quad \alpha_n = \alpha_{n-1} + \mu_n(z_n^2 - \alpha_{n-1}). \tag{8.25}$$

При оценке коэффициента корреляции центрированных случайных величин Z и Y с одинаковыми дисперсиями можно взять $J_1(\alpha, Z, Y) = M[(\alpha Z - Y)^2]$, тогда

$$\bar{\beta}_n = (\alpha_{n-1} z_n - y_n) z_n \quad \text{и} \quad \alpha_n = \alpha_{n-1} - \mu_n(\alpha_{n-1} z_n - y_n) z_n. \tag{8.26}$$

Обобщением последней задачи является оптимизация линейной оценки $Y = \bar{\alpha}^T \bar{Z}$, например, оптимизация линейного прогноза. В этом случае $J_1(\bar{\alpha}, \bar{Z}, Y) = M[(\bar{\alpha}^T \bar{Z} - Y)^2]$,

$$\bar{\beta}_n = (\bar{\alpha}_{n-1}^T Z_n - y_n) Z_n \quad \text{и}$$
$$\bar{\alpha}_n = \bar{\alpha}_{n-1} - \mu_n(\bar{\alpha}_{n-1}^T \bar{z}_n - y_n)\bar{z}_n. \tag{8.27}$$

Оценка квантилей случайных величин также может быть выполнена с помощью ПГ алгоритмов.

Случай 2. Иногда критерий качества выражается через вероятность события A, например, через вероятность правильного обнаружения. Этот случай можно свести к предыдущему, так как вероятность может быть выражена через математическое ожидание: $P(A) = M[\chi_A]$, где χ_A – индикатор события A ($\chi_A = 1$, если A произошло, и

$\chi_A = 0$, если A не произошло). Тогда реализациями являются оценки вероятности P(A) по частоте события A в каждом отдельном испытании (0 или 1). Для улучшения оценок можно использовать относительные частоты в группах из нескольких испытаний. При ненаблюдаемости события A иногда удается ввести вспомогательный функционал, выражаемый через параметры, влияющие на P(A). Например, через отношение сигнал/шум в задачах обнаружения.

Отметим, что вместо ПГ (8.23) часто можно взять

$$\bar{\beta}_n = \varphi(\nabla g(\alpha_{n-1}, Z_n)), \tag{8.28}$$

где φ – векторная функция той же размерности, что и ∇g. При этом φ может выбираться из широкого класса функций. Требуется, чтобы условие псевдоградиентности сохранялось. В частности, можно брать симметричные функции. Очень простые и в то же время хорошо сходящиеся алгоритмы получаются при знаковой функции φ

$$\varphi(\bar{a}) = \varphi(a_1, \ldots, a_n) = sign(a_1, \ldots, a_n) = (sign(a_1), \ldots sign(a_n))^T, \tag{8.29}$$

при которой компоненты $\bar{\alpha}_n$ в (8.19) отличаются от компонент $\bar{\alpha}_{n-1}$ на $\pm \mu_n$.

9. Растровая и векторная графика

9.1. Способы представления изображений в памяти ЭВМ

Формальное определение ***компьютерная (машинная) графика*** – это создание, хранение и обработка моделей объектов и их изображений с помощью ЭВМ. Под ***интерактивной*** компьютерной графикой понимают раздел компьютерной графики, изучающий вопросы динамического управления со стороны пользователя содержанием изображения, его формой, размерами и цветом на экране с помощью интерактивных устройств взаимодействия.

Под ***компьютерной геометрией*** понимают математический аппарат, применяемый в компьютерной графике.

Необходимо отметить следующую отличительную черту компьютерных изображений. **Изображения, которые мы встречаем в нашей повседневной жизни, реальные картины природы, можно бесконечно детализировать, выявлять все новые цвета и оттенки**. **Изображения, хранящиеся в памяти компьютера, независимо от способа их получения и представления, всегда являются усеченной моделью картины реального мира**. Их детализация возможна лишь с той степенью, которая была заложена при их создании или получении, и их цветовая гамма будет не шире заранее оговоренной.

Одно и то же изображение может быть представлено в памяти ЭВМ двумя принципиально различными способами и получено два различных типа изображения**: растровое и векторное**. Рассмотрим подробнее эти способы представления изображений, выделим их основные параметры и определим их достоинства и недостатки.

Что такое растровое изображение?

Возьмём фотографию (например, см. рис. 9.1). Конечно, она тоже состоит из маленьких элементов, но будем считать, что отдельные элементы мы рассмотреть не можем. **Она представляется для нас, как реальная картина природы.**

Рис. 9.1. Исходное изображение

Теперь разобьём это изображение на маленькие квадратики (маленькие, но всё-таки чётко различимые), и каждый квадратик закрасим цветом, преобладающим в нём (на самом деле программы при оцифровке генерируют некий «средний» цвет, т. е. если у нас была одна чёрная точка и одна белая, то квадратик будет иметь серый цвет).

Как мы видим, изображение стало состоять из конечного числа квадратиков определённого цвета. Эти квадратики называют *pixel* (от *PICture ELemen*t) – пиксел или пиксель.

Теперь каким-либо методом занумеруем цвета. Конкретная реализация этих методов нас пока не интересует. Для нас сейчас важно то, что каждый пиксель на рисунке стал иметь определённый цвет, обозначенный цифрой (рис. 9.2).

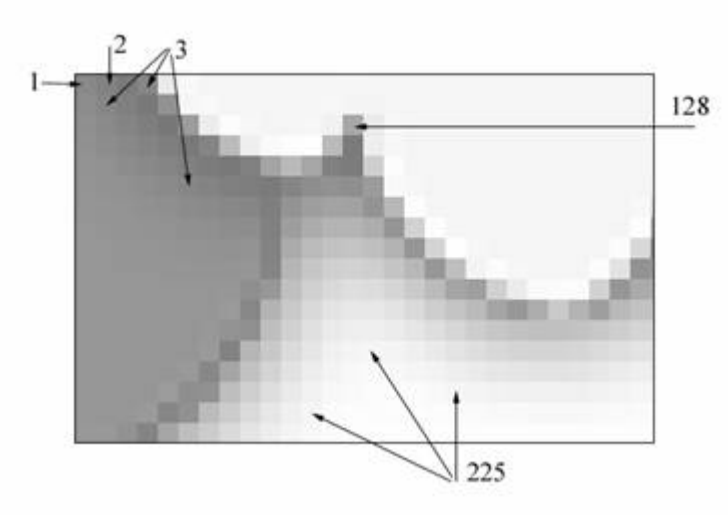


Рис. 9.2. Фрагмент оцифрованного изображения и номера цветов
Теперь пойдём по порядку (слева направо и сверху вниз) и будем в строчку выписывать номера цветов встречающихся пикселей. Получится строка примерно следующего вида:

1 2 8 3 212 45 67 45 127 4 78 225 34 ...

Вот эта строка и есть наши оцифрованные данные. Теперь мы можем сжать их (так как несжатые графические данные обычно имеют достаточно большой размер) и сохранить в файл.

Итак, **под *растровым* (bitmap, raster) понимают способ представления изображения в виде совокупности отдельных точек (пикселей) различных цветов или оттенков.** Это наиболее простой способ представления изображения, ибо **таким образом видит наш глаз**.

Достоинством такого способа является возможность получения фотореалистичного изображения высокого качества в различном цветовом диапазоне. Недостатком – высокая точность и широкий цветовой диапазон требуют увеличения объема файла для хранения изображения и оперативной памяти для его обработки.

Для *векторной* графики характерно разбиение изображения на ряд графических примитивов – точки, прямые, ломаные, дуги, полигоны. Таким образом, появляется возможность хранить не все точки изображения, а координаты узлов примитивов и их свойства (цвет, связь с другими узлами и т. д.).

Вернемся к изображению на рис.9.1. Взглянем на него по-другому. На изображении легко можно выделить множество простых объектов — отрезки прямых, ломанные, эллипс, замкнутые кривые. Представим себе, что пространство рисунка существует в некоторой координатной системе. Тогда **можно описать это изображение, как совокупность простых объектов, вышеперечисленных типов, координаты узлов которых заданы вектором относительно точки начала координат** (рис. 9.3).

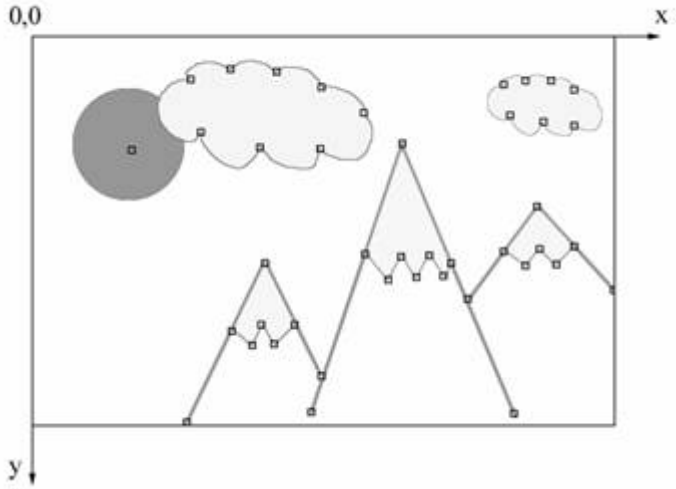


Рис. 9.3. Векторное изображение и узлы его примитивов

Проще говоря, **чтобы компьютер нарисовал прямую, нужны координаты двух точек, которые связываются по кратчайшей**

прямой. Для дуги задается радиус и т. д. Таким образом, **векторная иллюстрация – это набор геометрических примитивов.**

Важной деталью является то, что объекты задаются независимо друг от друга и, следовательно, могут перекрываться между собой.

При использовании ***векторного представления изображение хранится в памяти как база данных описаний примитивов***. **Основные графические примитивы, используемые в векторных графических редакторах: точка, прямая, кривая Безье, эллипс (окружность), полигон (прямоугольник). Примитив строится вокруг его узлов (nodes). Координаты узлов задаются относительно координатной системы макета**. А изображение будет представлять из себя массив описаний – нечто типа:

отрезок (20,20-100,80);

```
окружность(50,40-30);
кривая_Безье (20,20-50,30-100,50).
```

Каждому узлу приписывается группа параметров, в зависимости от типа примитива, которые задают его геометрию относительно узла. Например, **окружность задается одним узлом и одним параметром – радиусом**. Такой набор параметров, которые играют роль коэффициентов и других величин в уравнениях и аналитических соотношениях объекта данного типа, называют **аналитической моделью примитива**. **Отрисовать примитив – значит построить его геометрическую форму по его параметрам согласно его аналитической модели.**

Векторное изображение может быть легко масштабировано без потери деталей, так как это требует пересчета сравнительно небольшого числа координат узлов. Другой термин – «object-oriented graphics».

Самой простой аналогией векторного изображения может служить **аппликация**. **Все изображение состоит из отдельных кусочков различной формы и цвета (даже части растра), «склеенных» между собой.** Понятно, что таким образом трудно получить фотореалистичное изображение, так как на нем сложно выделить конечное число примитивов, однако существенными достоинствами

векторного способа представления изображения, по сравнению с растровым, являются:

- векторное изображение может быть легко масштабировано без потери качества, так как это требует пересчета сравнительно небольшого числа координат узлов;
- графические файлы, в которых хранятся векторные изображения, имеют существенно меньший, по сравнению с растровыми, объем (порядка нескольких килобайт).

Сферы применения векторной графики очень широки. В полиграфике – от создания красочных иллюстраций до работы со шрифтами. Все, что мы называем машинной графикой, 3D-графикой, графическими средствами компьютерного моделирования и САПР – все это сферы приоритета векторной графики, так как эти ветви дерева компьютерных наук рассматривают изображение **исключительно с позиции его математического представления.**

Как видно, векторным можно назвать только способ описания изображения, а само изображение для нашего глаза всегда растровое. Таким образом, **задачами векторного графического редактора являются растровая прорисовка графических примитивов и предоставление пользователю сервиса по изменению параметров этих примитивов. Все изображение представляет собой базу данных примитивов и параметров макета (размеры холста, единицы измерения и т. д.). Отрисовать изображение – значит выполнить последовательно процедуры прорисовки всех его деталей.**

Для уяснения разницы между растровой и векторной графикой приведем простой пример. Вы решили отсканировать Вашу фотографию размером 10×15 см чтобы затем обработать и распечатать на цветном принтере. Для получения приемлемого качества печати необходимо разрешение не менее 300 dpi. Считаем:

10 см = 3,9 дюйма; 15 см = 5,9 дюймов.

По вертикали: 3,9 * 300 = 1170 точек.

По горизонтали: 5,9 * 300 = 1770 точек.

Итак, число пикселей растровой матрицы 1170 * 1770 = 2 070 900.

Теперь решим, сколько цветов мы хотим использовать. Для черно-белого изображения используют обычно 256 градаций серого цвета для каждого пикселя, или 1 байт. Получаем, что для хранения нашего изображения надо 2 070 900 байт или 1,97 Мб.

Для получения качественного цветного изображения надо не менее 256 оттенков для каждого базового цвета. В модели RGB соответственно их 3: красный, зеленый и синий. Получаем общее количество байт – 3 на каждый пиксел. Соответственно, размер хранимого изображения возрастает в три раза и составляет 5,92 Мб.

Для создания макета для полиграфии фотографии сканируют с разрешением 600 dpi, следовательно, размер файла вырастает еще вчетверо.

С другой стороны, если изображение состоит из простых объектов, то для его хранения в векторном виде необходимо не более нескольких килобайт.

9.1.1. Классификация ПО компьютерной графики

На рис. 9.4. приведена классификация ПО, используемого при создании, редактировании и публикации изображений.

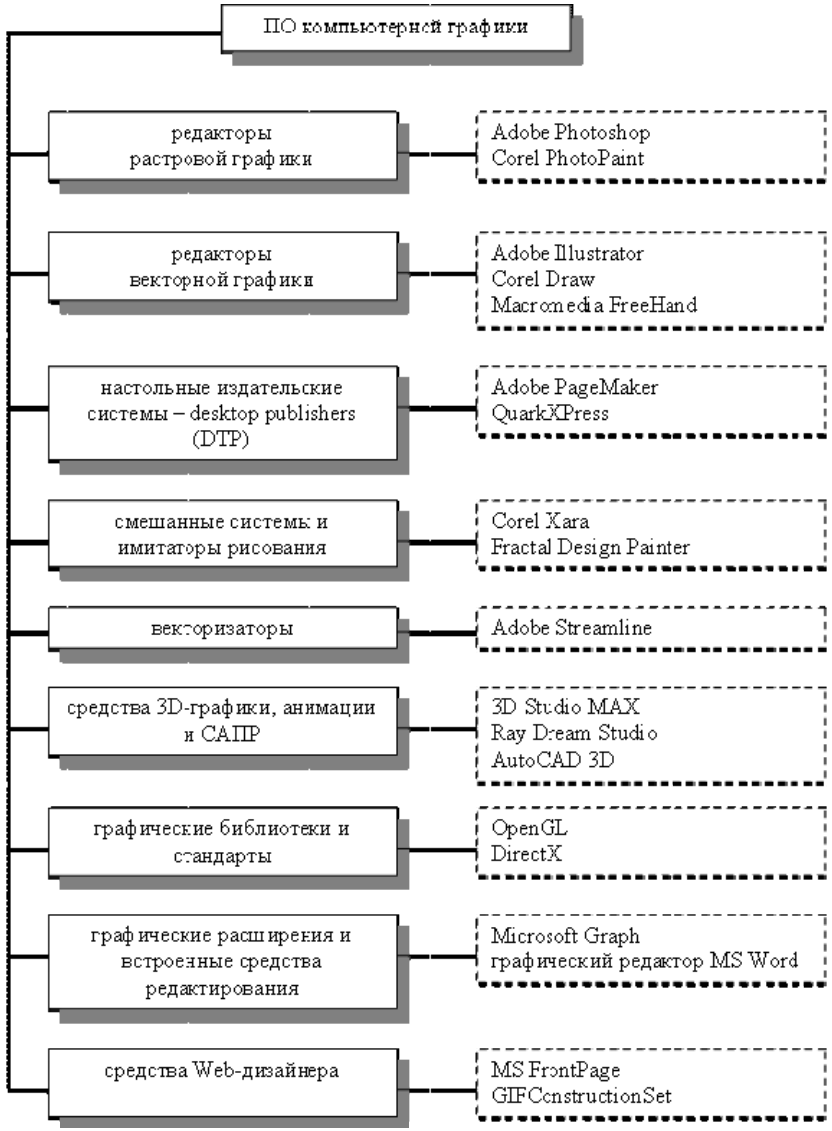


Рис. 9.4. Классификация ПО компьютерной графики

9.2. Параметры растровых изображений

Как уже говорилось, **растровое изображение представляется в памяти ЭВМ в виде матрицы отдельных пикселей.** В этой связи возникает вопрос о том, каково должно быть число этих пикселей и какое число бит отводится на один пиксель, т. е. каковы основные параметры растрового изображения – **разрешение и глубина цвета**.

Разрешение (resolution) — **это степень детализации изображения, число пикселей (точек), отводимых на единицу площади.** Поэтому имеет смысл говорить о разрешении изображения только применительно к какому-либо устройству ввода или вывода изображения. Например, **пока имеется обычная фотография на твердом носителе, нельзя сказать о ее разрешении. Но как только мы попытаемся ввести эту фотографию в компьютер через сканер, нам необходимо будет определить разрешение оригинала, т. е. указать количество точек, считываемых сканером с одного квадратного дюйма.**

Поскольку изображение можно рассматривать применительно к различным устройствам, то **следует различать**:

- разрешение оригинала;
- разрешение экранного изображения;
- разрешение печатного изображения.

Разрешение оригинала. Разрешение оригинала используется при вводе изображения в компьютер и измеряется в точках на дюйм (*dots per inch - dpi*). Установка разрешения оригинала зависит от требований, предъявляемых к качеству изображения и размеру файла. В общем случае действует правило: **чем выше требования к качеству, тем выше должно быть разрешение оригинала**.

Разрешение экранного изображения. Для экранных копий изображения элементарную точку растра принято называть пикселом (*pixel*). Для измерения разрешения экранного изображения, кроме dpi, используют *ppi* (*pixel per inch*). Размер пиксела варьируется в

зависимости от выбранного экранного разрешения (из диапазона стандартных значений), разрешения оригинала и масштаба отображения.

Мониторы для обработки изображений с диагональю 20–21 дюйм (профессионального класса), как правило, обеспечивают стандартные экранные разрешения 640×481 800×600, 1024×768,1280×1024, 1600×1200, 1600×1280, 1920×1200, 1920×1600 точек. **Расстояние между соседними точками люминофора у качественного монитора составляет 0,22–0,25 мм.**

Для экранной копии достаточно разрешения 72 dpi, для распечатки на цветном или лазерном принтере – 150–200 dpi, для вывода на фотоэкспонирующем устройстве – 200–300 dpi. Установлено эмпирическое правило, что при распечатке величина разрешения оригинала должна быть в 1,5 раза больше, чем линиатура растра устройства вывода. В случае если твердая копия будет увеличена по сравнению с оригиналом, эти величины следует умножить на коэффициент масштабирования.

Разрешение печатного изображения и понятие линиатуры. Размер точки растрового изображения на твердой копии (бумаге, пленке и т. д.) зависит от примененного метода и параметров растрирования оригинала. **При растрировании на оригинал как бы накладывается сетка линий, ячейки которой образуют элемент растра. Частота сетки растра измеряется числом линий на дюйм (*lines per inch — lpi*) и называется *линиатурой*.**

Рассмотрим простейшие методы растрирования черно-белого оригинала. Размер точки растра рассчитывается для каждого элемента и зависит от интенсивности тона в данной ячейке. Чем больше интенсивность, тем плотнее заполняется элемент растра, т. е. если в ячейку попал абсолютно черный цвет, размер точки растра совпадет с размером элемента растра. В этом случае говорят о 100 % заполняемости. Для абсолютно белого цвета значение заполняемости составит 0 %. На практике заполняемость элемента на отпечатке обычно составляет от 3 до 98 %, при этом все точки растра имеют одинаковую оптическую плотность, в идеале приближающуюся к абсолютно черному цвету. Иллюзия более темного тона создается за счет увеличения размеров точек и, как следствие, сокращения пробельного поля между ними при одинаковом расстоянии между

центрами элементов растра. Такой метод называют ***растрированием с амплитудной модуляцией*** *(АМ)*.

Существует и метод ***растрирования с частотной модуляцией*** *(ЧМ)*, когда **интенсивность тона регулируется изменением расстояния между соседними точками одинакового размера**. Таким образом, при частотно-модулированном растрировании в ячейках растра с разной интенсивностью тона находится разное число точек. Изображения**, растрированные ЧМ-методом, выглядят более качественно, так как размер точек минимален и, во всяком случае, существенно меньше, чем средний размер точки при АМ-растрировании.** Еще более повышает качество изображения разновидность ЧМ-метода, называемая ***стохастическим растрированием***. В этом случае рассчитывается число точек, необходимое для отображения требуемой интенсивности тона в ячейке растра. Затем эти точки располагаются внутри ячейки на расстояниях, вычисленных квазислучайным методом (**на самом деле используется специальный математический алгоритм**), т. е. **регулярная структура растра внутри ячейки, как и на изображении в целом, вообще отсутствует.** Поэтому при стохастическом ЧМ-растрировании теряет смысл понятие линиатуры растра, имеет значение лишь разрешающая способность устройства вывода. Такой способ требует больших затрат вычислительных ресурсов и высокой точности.

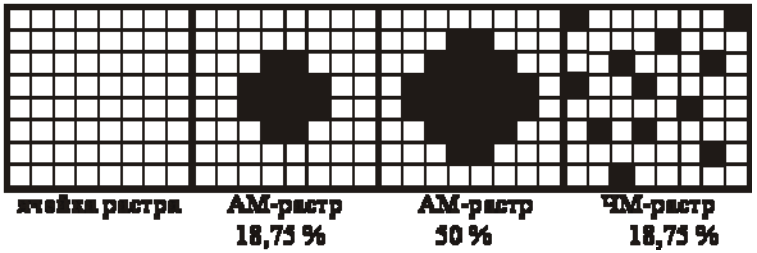


Рис. 9.5. Примеры амплитудной и частотной модуляции растра
Задание**: зная, что размер экрана в пикселях 800×600, а разрешение 72 ppi, установить реальные размеры экрана в сантиметрах.**

Глубина цвета (color depth) — это число бит, используемых для представления каждого пикселя изображения, определяемое цветовым или тоновым диапазоном.

Цветовой (тоновый) диапазон (color range) — это максимальное число цветов, используемых при создании изображения.

9.3. Представление цвета в компьютере

Понятия света и цвета в компьютерной графике являются основополагающими. **Свет можно рассматривать двояко**: **либо как поток частиц различной энергии, либо как поток электромагнитных волн.**

Понятие цвета тесно связано с тем, как человек воспринимает свет. Можно сказать, что **ощущение света формируется человеческим мозгом в результате анализа светового потока, попадающего на сетчатку глаз.**

Источник или объект является ***ахроматическим,*** если наблюдаемый свет содержит все видимые длины волн в приблизительно равных количествах. Ахроматическими цветами являются белый, черный, градации серого цвета. Например, белыми выглядят объекты ахроматически отражающие более 80 % света белого источника, а черными – менее 3 %.

Если воспринимаемый свет содержит длины волн в неравных количествах, то он называется ***хроматическим***.

Считается, что в глазе человека существует три группы цветовых рецепторов (колбочек), каждая из которых чувствительна к определенной длине световой волны. Каждая группа формирует один из трех ***основных цветов***: красный, зеленый, синий.

Если длины волн светового потока сконцентрированы у верхнего края видимого спектра (около 700 Нм), то свет воспринимается как красный. Если длины волн сконцентрированы у нижнего края видимого спектра (около 400 Нм), то свет воспринимается как синий. Если длины волн сконцентрированы в середине видимого спектра (около 550 Нм), то свет воспринимается как зеленый.

С помощью экспериментов, построенных на этой гипотезе, были получены кривые реакции глаза, показанные на рис.9.6.

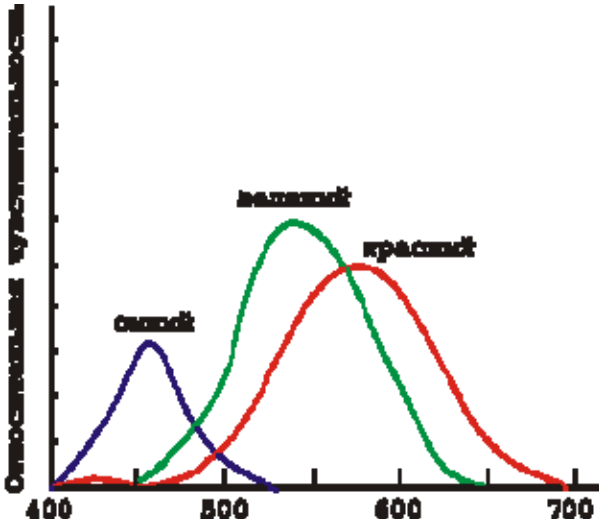


Рис. 9.6. Кривые реакции глаза

Физические характеристики светового потока определяются параметрами *мощности*, *яркости* **и** *освещенности*. **Визуальные параметры ощущения цвета характеризуются** *светлотой*, *насыщенностью* **и** *цветовым тоном*.

Светлота – это различимость участков, сильнее или слабее отражающих свет. Минимальную разницу между яркостью различимых по светлоте объектов называют ***порогом***.

Насыщенность цвета показывает, насколько данный цвет отличается от монохроматического («чистого») излучения того же светового тона. Насыщенность характеризует степень ослабления (разбавления) данного цвета белым и позволяет отличать розовый от красного, голубой от синего.
Цветовой тон позволяет различать основные цвета, такие, как красный, зеленый, синий.

9.3.1. Цветовые модели

Как видим из вышеизложенного, описание цвета может опираться на составление любого цвета на основе основных цветов или на такие понятия, как светлота, насыщенность, цветовой тон. Применительно к компьютерной графике описание цвета также должно учитывать

специфику аппаратуры для ввода/вывода изображений. В связи с необходимостью описания различных физических процессов воспроизведения цвета были разработаны различные цветовые модели. Цветовые модели позволяют с помощью математического аппарата описать определенные цветовые области спектра. Цветовые модели описывают цветовые оттенки с помощью смешивания нескольких основных цветов.

Основные цвета разбиваются на оттенки по яркости (от темного к светлому), и каждой градации яркости присваивается цифровое значение (например, самой темной – 0, самой светлой – 255). Считается, что в среднем человек способен воспринимать около 256 оттенков одного цвета. Таким образом, любой цвет можно разложить на оттенки основных цветов и обозначить его набором цифр – цветовых координат.

Таким образом, при выборе цветовой модели можно определять трехмерное цветовое координатное пространство, внутри которого каждый цвет представляется точкой. Такое пространство называется пространством цветовой модели.

Профессиональные графические программы обычно позволяют оперировать с несколькими цветовыми моделями, большинство из которых создано для специальных целей или особых типов красок: CMY, CMYK, CMYK256, RGB, HSB, HLS, L*a*b, YIQ, Grayscale (Оттенки серого) и Registration color. Некоторые из них используются редко, диапазоны других перекрываются.

Цветовая модель RGB. В основе одной из наиболее распространенных цветовых моделей, называемой RGB моделью, лежит воспроизведение любого цвета путем сложения трех основных цветов: красного (Red), зеленого (Green) и синего (Blue). Каждый канал - R, G или B имеется свой отдельный параметр, указывающий на количество соответствующей компоненты в конечном цвете. Например: (255, 64, 23) – цвет, содержащий сильный красный компонент, немного зелёного и совсем немного синего. Естественно, что этот режим наиболее подходит для передачи богатства красок окружающей природы. Но он требует и больших расходов, так как глубина цвета тут наибольшая – 3 канала по 8 бит на каждый, что дает в общей сложности 24 бита.

Поскольку в RGB модели происходит сложение цветов, то она называется ***аддитивной*** (additive). Именно на такой модели построено воспроизведение цвета современными мониторами.

Цветовым пространством RGB модели является единичный куб.

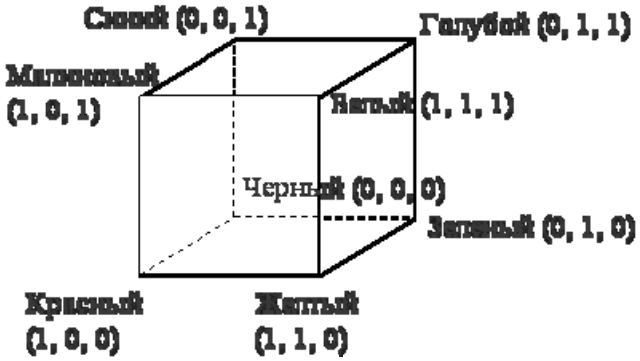


Рис. 9.7. Цветовое пространство RGB модели

Цветовые модели CMY и CMYK. Модель CMY использует также три основных цвета: Cyan (голубой), Magenta (пурпурный, или малиновый) и Yellow (желтый). Эти цвета описывают отраженный от белой бумаги свет трех основных цветов RGB модели. Поэтому можно описать соотношения между RGB и CMY моделями следующим образом:

$$\begin{pmatrix} C \\ M \\ Y \end{pmatrix} = \begin{pmatrix} 1 \\ 1 \\ 1 \end{pmatrix} - \begin{pmatrix} R \\ G \\ B \end{pmatrix} \qquad \begin{pmatrix} R \\ G \\ B \end{pmatrix} = \begin{pmatrix} 1 \\ 1 \\ 1 \end{pmatrix} - \begin{pmatrix} C \\ M \\ Y \end{pmatrix}.$$

Модель CMY является ***субтрактивной*** (основанной на вычитании) цветовой моделью. Как уже говорилось, в CMY-модели описываются

цвета на белом носителе, т. е. краситель, нанесенный на белую бумагу, вычитает часть спектра из падающего белого света. Например, на поверхность бумаги нанесли голубой (Cyan) краситель. Теперь красный свет, падающий на бумагу, полностью поглощается. Таким образом, голубой носитель вычитает красный свет из падающего белого.

Такая модель наиболее точно описывает цвета при выводе изображения на печать, т. е. в полиграфии.

Поскольку для воспроизведения черного цвета требуется нанесение трех красителей, а расходные материалы дороги, использование CMY-модели является не эффективным. Дополнительный фактор, не добавляющий привлекательности CMY-модели, – это появление нежелательных визуальных эффектов, возникающих за счет того, что при выводе точки три базовые цвета могут ложиться с небольшими отклонениями. Поэтому к базовым трем цветам CMY-модели добавляют черный (blacK) и получают новую цветовую модель CMYK.

Для перехода из модели CMY в модель CMYK иногда используют следующее соотношение:

$$K = \min(C, M, Y);$$

$$C = C - K;$$

$$M = M - K;$$

$$Y = Y - K.$$

Соотношения преобразования RGB в CMY и CMY в CMYK-модель верны лишь в том случае, когда спектральные кривые отражения для базовых цветов не пересекаются. Поэтому в общем случае можно сказать, что существуют цвета, описываемые в RGB-модели, но не описываемые в CMYK-модели.

Существует также модель CMYK256, которая используется для более точной передачи оттенков при качественной печати изображений.

Цветовые модели HSV и HLS. Рассмотренные модели ориентированы на работу с цветопередающей аппаратурой и для некоторых людей

неудобны. Поэтому модели HSV, HLS опираются на интуитивные понятия тона насыщенности и яркости.

В цветовом пространстве модели HSV (Hue, Saturation, Value), иногда называемой HSB (Hue, Saturation, Brightness), используется цилиндрическая система координат, а множество допустимых цветов представляет собой шестигранный конус, поставленный на вершину.

Основание конуса представляет яркие цвета и соответствует $V = 1$. Однако цвета основания $V = 1$ не имеют одинаковой воспринимаемой интенсивности. Тон (H) измеряется углом, отсчитываемым вокруг вертикальной оси OV. При этом красному цвету соответствует угол 0°, зелёному – угол 120° и т. д. Цвета, взаимно дополняющие друг друга до белого, находятся напротив один другого, т. е. их тона отличаются на 180°. Величина S изменяется от 0 на оси OV до 1 на гранях конуса.

Конус имеет единичную высоту ($V = 1$) и основание, расположенное в начале координат. В основании конуса величины H и S смысла не имеют. Белому цвету соответствует пара $S = 1$, $V = 1$. Ось OV ($S = 0$) соответствует ахроматическим цветам (серым тонам).

Процесс добавления белого цвета к заданному можно представить как уменьшение насыщенности S, а процесс добавления чёрного цвета – как уменьшение яркости V. Основанию шестигранного конуса соответствует проекция RGB куба вдоль его главной диагонали.

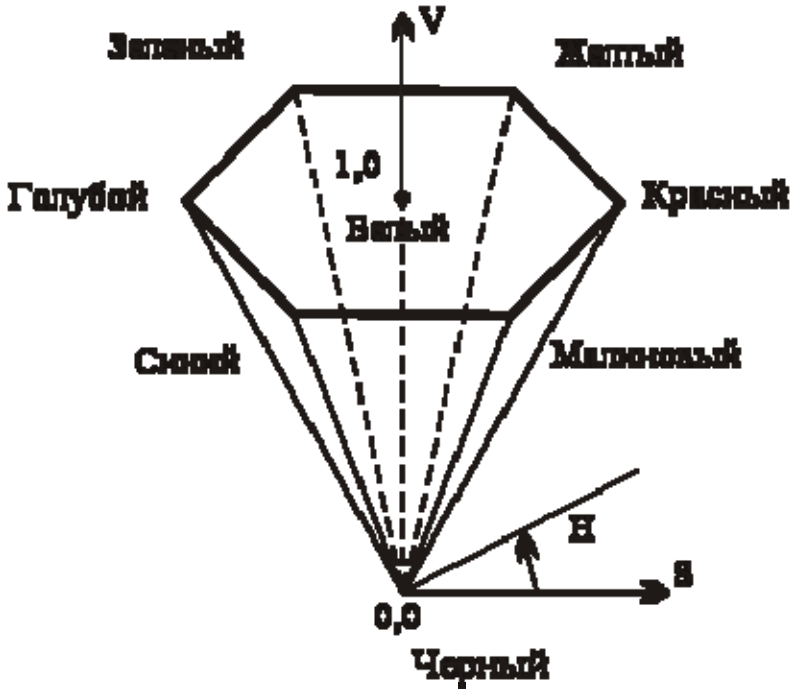


Рис. 9.8. Цветовое пространство HSV модели

Еще одним примером системы, построенной на интуитивных понятиях тона насыщенности и яркости, является система HLS (Hue, Lightness, Saturation). Здесь множество всех цветов представляет собой два шестигранных конуса, поставленных друг на друга (основание к основанию).

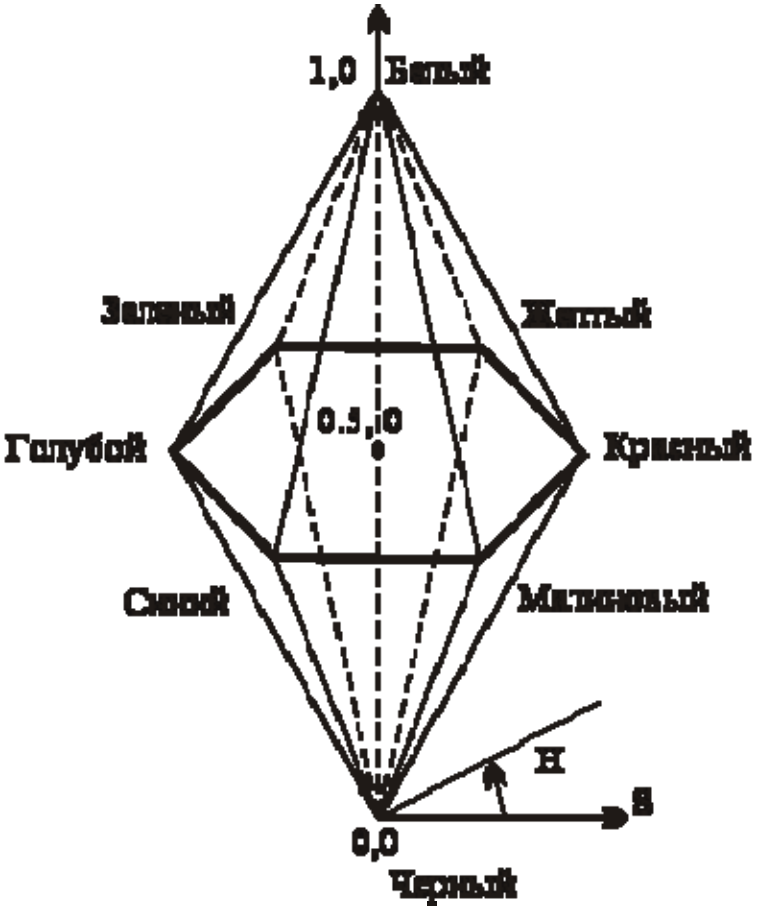


Рис. 9.9. Цветовое пространство HLS-модели

Полноцветные и индексированные изображения. **Как мы увидели, цвета пикселов можно определять, явно задавая несколько параметров цвета. Например, в RGB-модели конечный цвет определяется тремя слагаемыми для трех основных цветов. Такой подход позволяет формировать так называемые** *полноцветные* **изображения.**

Второй подход заключается в том, что в первой части файла, хранящего изображение, хранится ***«палитра»***, в которой с помощью

одной из цветовых моделей кодируются цвета, присутствующие на изображении. А вторая часть, которая непосредственно описывает пикселы изображения, фактически состоит из индексов в палитре. Изображения, формируемые таким способом, называются изображениями с ***индексированной палитрой***.

Частным случаем индексированного изображения является черно-белое изображение. В подобном изображении могут быть только 2 цвета - чёрный и белый, кодируемые соответственно 0 и 1. Глубина изображения составляет в данном случае 1 бит. Эта глубина очень плохо подходит к представлению фотореалистичных образов и применяется лишь для специализированных изображений.

Достоинством палитры является возможность существенно сократить размер файла с изображением. Недостатком является возможность потери цветов при ограниченном размере палитры. Обычно размер палитры составляет до 256-ти цветов.

9.3.2. Системы управления цветом

Ситуация, когда дизайнерам и полиграфистам приходится работать в разных цветовых пространствах, приводит к возникновению ошибок в цветопередаче на этапе перехода от одной цветовой модели к другой. Альтернативой такого подхода является использование так называемой аппаратно независимой системы управления цветом (color managing system, CMS). **Суть этой технологии состоит в том, чтобы независимо от программного обеспечения цвет передавался от одного этапа обработки (например, сканирования) к другой (печати) без искажений**. Таким образом, обеспечивается видимая однородность цветового пространства для всех периферийных устройств и приложений, работающих в системе. Изначальная поддержка этой технологии на платформе Macintosh (CMS ColorSync) была долгое время причиной предпочтения этой платформы специалистами в области графики. В последнее время много говорится о работе с этой технологией корпорации Microsoft.

CMS содержит набор объективных параметров, обязательных для всех устройств при обмене цветовыми данными. Универсальность CMS достигается введением трех типов переменных, каждая из которых управляет цветом на своем уровне.

Цветовая гамма. Любой тип устройства имеет свою ***цветовую гамму***, область, которая всегда меньше, чем цветовой охват практически любой цветовой модели. CMS управляет преобразованием цвета между различными цветовыми моделями с учетом цветовой гаммы конкретных устройств.

Профиль. Каждое устройство воспроизводит цвета особым образом, что зависит от технических и программных решений, принятых изготовителем. Для согласования отображения цветов на различных устройствах они должны иметь собственный ***профиль***, описывающий различие в представлении цвета между устройством и определенной цветовой моделью.

Международный консорциум по цвету (ICC – International Color Consortium) установил стандарт на параметры описания характеристик воспроизведения цвета.

Калибровка. Даже устройства одной модели от одного производителя имеют отличия в реализации профиля ICC, обусловленные допусками при изготовлении, условиями эксплуатации, внешними помехами. Поэтому CMS обычно включают в себя средства калибровки, т. е. средства настройки конкретного экземпляра в соответствии с требованиями профиля ICC. Средства калибровки бывают аппаратно-программными и чисто программными.

Не существует идеальной CMS, одинаково пригодной для всех устройств, одинаково работающей на всех платформах и программных средствах. Наиболее удачными можно считать CMS, реализованные на уровне операционной системы. В операционной системе Windows используется модуль CMS фирмы Kodak, называемый Color Matching Module. Однако поддержка со стороны производителей пока явно недостаточна.

Из CMS, являющихся внешними по отношению к операционным системам, наибольшее распространение получили программы фирм, давно работающих в области цветной фотографии, печати, цифровых графических технологий: Agfa Foto Tune, Kodak DayStar Color Match.

Что же делать, если CMS для вас еще недоступна, а адекватность восприятия цветов сохранить необходимо? Тут существует две альтернативы: постоянная калибровка периферийного оборудования

(сканера, монитора и т. д.) или **использование специальных атласов цветов (color sample card)**. Первый путь довольно дорогой и требующий участия специалистов. **Атлас цветов представляет собой совокупность заранее сформированных стандартизированных оттенков цветов, сведенных в упорядоченную таблицу, которые можно использовать напрямую по мере необходимости. Наиболее распространенными на сегодняшний день являются атласы фирмы Pantone.**

9.4. Графические файловые форматы

Как уже говорилось ранее, при хранении растровых изображений, как правило, приходится иметь дело с файлами большого размера. В этой связи важной задачей является выбор соответствующего формата файла.

Форматов графических файлов существует великое множество и выбор приемлемого отнюдь не является тривиальной задачей. Для облегчения выбора воспользуемся классификациями.

1. **По типу хранимой графической информации:**

- растровые (TIFF, GIF, BMP, JPEG);
- векторные (AI, CDR, FH7, DXF);
- смешанные/универсальные (EPS, PDF).

Следует учитывать, что файлы практически любого векторного формата позволяют хранить в себе и растровую графику. Однако часто это приводит к искажениям в цветопередаче, поэтому если изображение не содержит векторных объектов, то предпочтительнее использовать растровые форматы.

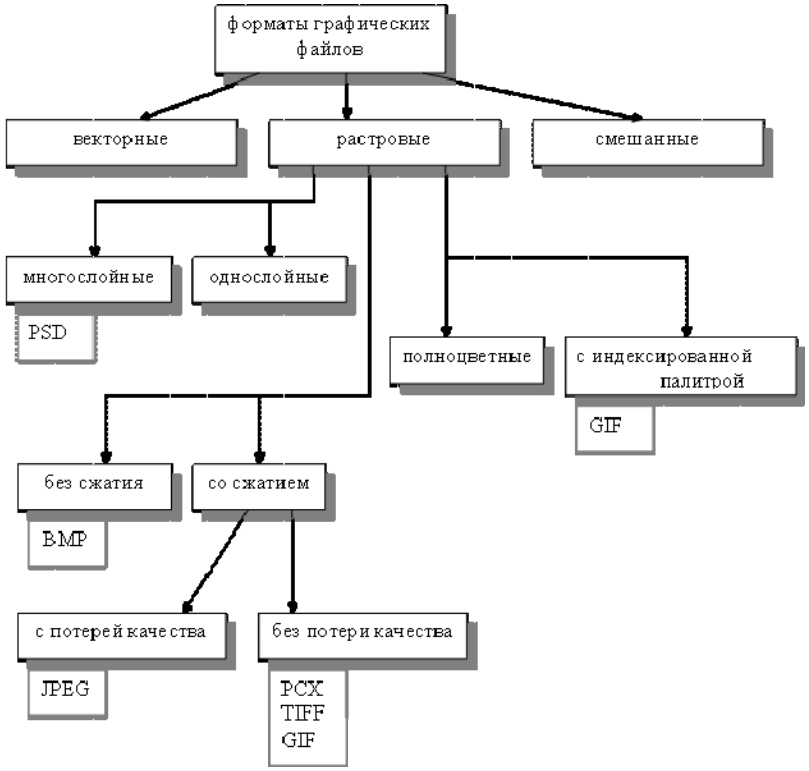


Рис. 9.10. Графические файловые форматы

Рассмотрим вкратце наиболее популярные форматы растровой графики.

TIFF

Предложенный компанией Aldus формат TIFF (Tagged Image File Format) на сегодняшний день ближе всех к статусу стандартного. Помимо прочих достоинств формат TIFF позволяет сохранять растровые изображения с компрессией без потери качества. Помимо традиционных цветов CMY формат поддерживает цветоделение с большим числом красок, в частности систему Hexahrome компании Pantone. Этот формат поддерживает сжатие без потери качества по алгоритму LWZ-компрессии. Наиболее предпочтителен для полиграфии. Принцип хранения данных основан на использовании

специальных маркеров (тэгов) в сочетании с битовыми последовательностями кусков растра.

GIF

Первая версия формата GIF (Graphics Interchange Format, «Формат для обмена графической информацией») была разработана в 1987 г. специалистами компьютерной сети CompuServe. Этот формат сочетает в себе редкий набор достоинств, неоценимых при той роли, которую он играет в WWW. Сам по себе формат содержит уже достаточно хорошо упакованные графические данные.

Как и у программ-архиваторов, степень сжатия графической информации в GIF сильно зависит от уровня ее повторяемости и предсказуемости, а иногда еще и от ориентации картинки; поскольку GIF сканирует изображение по строкам, то, к примеру, плавный переход цветов (градиент), направленный сверху вниз, сожмется куда лучше, чем тех же размеров градиент, ориентированный слева направо, а последний – лучше, чем градиент по диагонали.

GIF может иметь любое количество цветов от двух до 256-ти, и если в изображении используется, скажем, 64 цвета (2^6), то для хранения каждого пиксела будет использовано ровно шесть бит и ни битом больше.

Изменив порядок следования данных в файле, создатели GIFa заставили картинку рисоваться не только сверху вниз, но и, если можно так выразиться, «с глубины к поверхности», – то есть становиться все четче и детальнее по мере подхода из сети новых данных.

Для этого файл с изображением тасуется при записи так, чтобы сначала шли все строки пикселов с номерами, кратными восьми (первый проход), затем четырем (второй проход), потом двум и, наконец, последний проход – все оставшиеся строки с нечетными номерами. Во время приема и декодирования такого файла каждый следующий проход заполняет «дыры» в предыдущих, постепенно приближая изображение к исходному состоянию. Поэтому такие изображения были названы чересстрочными (interlaced).

Другой полезной возможностью формата является использование прозрачности.

Формат может быть использован для создания анимационных изображений. Для создания таких файлов используется утилита GIFConstractionSet.

BMP

Формат BMP (от слова bitmap) был создан компанией Microsoft и широко используется в операционных системах семейства Windows для растровой графики. Вам необходимо записать изображение в этом формате, если вы хотите использовать его в качестве фона вашего рабочего стола. Хотя в этом формате может применяться компрессия, большинство программ ее не используют.

В файлах BMP информация о цвете каждого пиксела кодируется 1, 4, 8, 16 или 24 бит (бит/пиксел).

JPEG (Joint Photographic Experts Group)

Строго говоря, JPEG называется не формат, а алгоритм сжатия, основанный не на поиске одинаковых элементов, как в RLE и LZW, а на разнице между пикселами. JPEG ищет плавные цветовые переходы в квадратах 9×9 пикселов. Вместо действительных значений JPEG хранит скорость изменения от пиксела к пикселу. Лишнюю с его точки зрения цветовую информацию он отбрасывает, усредняя некоторые значения. Чем выше уровень компрессии, тем больше данных отбрасывается и тем ниже качество. Используя JPEG, можно получить файл в 10–500 раз меньше, чем BMP. Формат аппаратно независим, полностью поддерживается на PC и Macintosh, однако он относительно нов и не понимается старыми программами (до 1995 г.). Из сказанного можно сделать следующие выводы. С помощью JPEG лучше сжимаются растровые картинки фотографического качества, чем логотипы или схемы – в них больше полутоновых переходов, среди же однотонных заливок появляются нежелательные помехи. Изображения с высоким разрешением (200–300 и более dpi) сжимаются с меньшими потерями, чем с низким (72–150 dpi), так как в каждом квадрате 9×9 пикселов переходы получаются более мягкие за счет того, что их (квадратов) в файлах высокого разрешения больше. В формате JPEG следует сохранять только конечный вариант работы,

потому что каждое пересохранение приводит ко все новым потерям (отбрасыванию) данных и превращению исходного изображения в кашу. Как это ни парадоксально, возможности алгоритма сжатия JPEG реализованы в формате JPEG не полностью.

PDF

Формат PDF (Portable Document Format) предложен фирмой Adobe как независимый от платформы формат, в котором могут быть сохранены и иллюстрации (векторные и растровые), и текст, причем со множеством шрифтов и гипертекстовых ссылок. Для достижения продекларированной в названии переносимости размер PDF-файла должен быть малым. Для этого используется компрессия (для каждого вида объектов применяется свой способ). Например, растровые изображения записываются в формате JPEG. Для работы с этим форматом компания Adobe выпустила пакет Acrobat. Бесплатная утилита Acrobat Reader позволяет читать документы и распечатывать их на принтере, но не дает возможности создавать или изменять их. Acrobat Distiller переводит в этот формат PostScript-файлы. Многие программы (Adobe PageMaker, CorelDraw, FreeHand) позволяют экспортировать свои документы в PDF, а некоторые – еще и редактировать графику, записанную в этом формате. Обычно в этом формате хранят документы, предназначенные только для чтения, но не для редактирования. Файл в формате PDF содержит все необходимые шрифты. Это удобно и позволяет не передавать шрифты для вывода (передача шрифтов не вполне законна с точки зрения авторского права).

PostScript

Это язык описания страниц, предназначеный для формирования изображений произвольной сложности и вывода их на печать. Для этого в языке имеется широкий набор графических операторов, используемых в произвольной комбинации. Все графические операторы языка, формирующие изображение, можно разделить на три группы. Это:

- **векторная графика**, позволяющая рисовать прямые линии, дуги, кривые произвольного размера, ориентации, ширины, закрашивать площади любого размера, формы, цвета; цвет для линий или заливок может задаваться в любом из цветовых пространств

языка; любой описанный на языке контур может быть границей клипирования изображения; контур клипирования задаёт границы рисуемого изображения;

- **работа с текстом** – для вывода текста произвольного размера в различных гарнитурах, размещая его с произвольной ориентацией в произвольном месте страницы; текст полностью интегрирован с графикой – все текстовые символы трактуются как графические фигуры и могут обрабатываться любым из графических операторов;

- **растровые изображения** позволяют выводить на листе сканированные рисунки или фотографии с масштабированием и ориентацией, источником растра может быть как текущий файл, содержащий программу на PostScripte, так и внешний; считывание цветовых слоев может вестись как из одного файла, так и из нескольких сепарированных. Изображение, описываемое на языке PostScript, никак не зависит от разрешающей способности выходного устройства и его цветовой глубины (числа цветов). Приближение к конкретным разрешающим возможностям выходного устройства – это процесс, не связанный с описанием изображения на языке PostScript, и выполняется для каждого выходного устройства по-своему. В этом и заключается устройство-независимость языка PostScript. Качество изображения определяется конкретным выходным устройством, его физическими ограничениями.

Формирование изображения на выходном устройстве является двухступенчатым процессом:

1. Приложение генерирует устройство независимое изображение на языке PostScript.

2. Система обработки изображения (интерпретатор) интерпретирует изображение (программу) и приближает его к характеристикам конкретного выходного устройства.

10. Растровые алгоритмы

Большинство графических устройств являются растровыми, представляя изображение в виде прямоугольной матрицы (сетки, целочисленной решетки) пикселей (растра), и большинство графических библиотек содержат внутри себя достаточное количество простейших растровых алгоритмов. На рис. 10.1 приведена система растровых алгоритмов.

Рис. 10.1. Классификация растровых алгоритмов

10.1. Алгоритмы растеризации

Прежде чем перейдем к непосредственному рассмотрению возможности перевода математического описания объекта (линии и пр.) в растровую форму, рассмотрим понятие связности.

Связность – возможность соединения двух пикселей растровой линией, т. е. последовательным набором пикселей. Возникает вопрос, когда пиксели (x_1, y_1) и (x_2, y_2) можно считать соседними. Для этого вводятся два понятия связности:

1. Четырехсвязность: пиксели считаются соседними, если либо их *x*-координаты, либо их *y* – координаты отличаются на единицу:

$$|x_1 - x_2| + |y_1 - y_2| \le 1;$$

2. Восьмисвязность: пиксели считаются соседними, если их *x*-координаты и *y*-координаты отличаются не более чем на единицу:

$$|x_1 - x_2| \le 1,\ |y_1 - y_2| \le 1.$$

На рис. 10.2 изображены четырехсвязная и восьмисвязная линии.

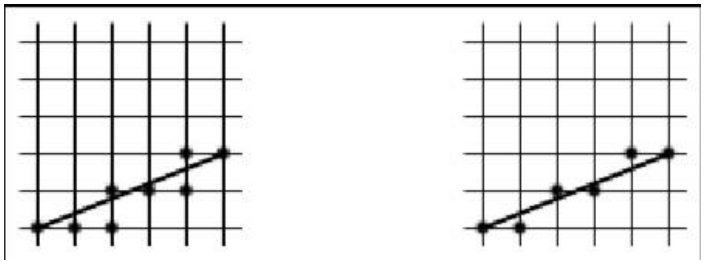


Рис. 10.2. Четырехсвязная и восьмисвязная линии

При переводе объектов в растровое представление существуют, алгоритмы, как использующие четырехсвязность, так использующие восьмисвязность.

10.1.1. Растровое представление отрезка. Алгоритм Брезенхейма

Рассмотрим задачу построения растрового изображения отрезка, соединяющего точки $A(x_a, y_a)$ и $B(x_b, y_b)$. Для простоты будем считать, что $0 \le y_b - y_a \le x_b - x_a$. Тогда отрезок описывается уравнением:

$$y = y_a + \frac{y_b - y_a}{x_b - x_a}(x - x_a),\ x \in [x_a, x_b] \text{ или } y = kx + b.$$

Отсюда получаем простейший алгоритм растрового представления отрезка:

```
     void line(int xa, int ya, int xb, int yb, int
color)
     {
     double k = ((double)(yb - ya)) / (xb - xa);
     double b = ya - k * xa;
     for (int x = xa; x <= xb; x++)
     putpixel(x, (int)(k * x + b), color);}
```

Вычислений значений функции $y = kx + b$ можно избежать, используя в цикле рекуррентные соотношения, так как при изменении *x* на 1 значение *y* меняется на *k*:

```
     void line(int xa, int ya, int xb, int yb, int
color)
     {
     double k = ((double)(yb - ya)) / (xb - xa);
     double y = ya;
     for (int x = xa; x <= xb; x++, y += k)
     putpixel(x, (int)y, color);
     }
```

Приведенные простейшие пошаговые алгоритмы построения отрезка имеют ряд недостатков:

1. Выполняют операции над числами с плавающей точкой, а желательно было бы работать с целочисленной арифметикой;
2. На каждом шаге выполняется операция округления, что также снижает быстродействие.

Эти недостатки устранены в следующем алгоритме Брезенхейма.

Как и в предыдущем случае, будем считать, что тангенс угла наклона отрезка принимает значение в диапазоне от 0 до 1. Рассмотрим *i*-й шаг алгоритма (рис. 10.3). На этом этапе пиксель P_{i-1} уже найден как ближайший к реальному отрезку. Требуется определить, какой из пикселов (T_i или S_i) будет установлен следующим.

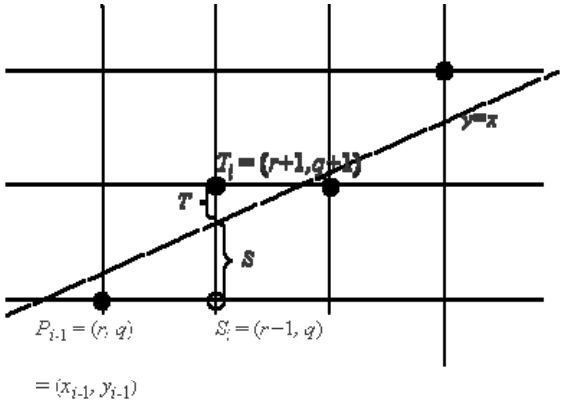


Рис. 10.3. i-й шаг алгоритма Брезенхейма

В алгоритме используется управляющая переменная d_i, которая на каждом шаге пропорциональна разности между S и T. Если $S < T$, то S_i ближе к отрезку, иначе выбирается T_i.

Пусть изображаемый отрезок проходит из точки (x_1, y_1) в точку (x_2, y_2). Исходя из начальных условий, точка (x_1, y_1) ближе к началу координат. Тогда перенесем оба конца отрезка с помощью преобразования $T(-x_1, -y_1)$, так чтобы первый конец отрезка совпал с началом координат. Начальной точкой отрезка стала точка (0, 0), конечной точкой – (dx, dy), где $dx = x_2 - x_1$, $dy = y_2 - y_1$ (рис. 10.4).

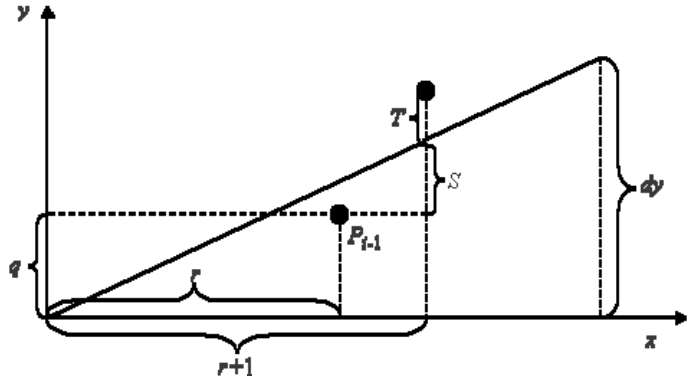


Рис. 10.4. Вид отрезка после переноса в начало координат

Уравнение прямой в этом случае будет иметь вид:

$$y = x \frac{dy}{dx}.$$

Обозначим координаты точки P_{i-1} после переноса через (r, q). Тогда $S_i = (r+1, q)$ и $T_i = (r+1, q+1)$.

Из подобия треугольников на рис. 10.4 можно записать, что

$$\frac{dy}{dx} = \frac{S+q}{r+1}.$$

Выразим S:

$$S = \frac{dy}{dx}(r+1) - q.$$

T можно представить как $T = 1 - S$. Используем предыдущую формулу

$$T = 1 - S = 1 - \frac{dy}{dx}(r+1) - q.$$

Найдем разницу $S - T$:

$$S - T = \frac{dy}{dx}(r+1) - q - 1 + \frac{dy}{dx}(r+1) - q = 2\frac{dy}{dx}(r+1) - 2q - 1.$$

Помножим левую и правую часть на dx:

$$dx(S - T) = 2\,dy(r+1) - 2q\,dx - dx = 2(r\,dy - q\,dx) + 2\,dy - dx.$$

Величина dx положительная, поэтому неравенство $dx(S - T) < 0$ можно использовать в качестве проверки при выборе S_i. Обозначим $d_i = dx(S - T)$, тогда

$$d_i = 2\,(r\,dy - q\,dx) + 2\,dy - dx.$$

Поскольку $r = x_{i-1}$ и $q = y_{i-1}$, то

$$d_i = 2\,x_{i-1}\,dy - 2\,y_{i-1}\,dx + 2\,dy - dx.$$

Прибавляя 1 к каждому индексу найдем d_{i+1}:

$$d_{i+1} = 2\,x_i\,dy - 2\,y_i\,dx + 2\,dy - dx.$$

Вычитая d_i из d_{i+1} получим

$$d_{i+1} - d_i = 2\,\mathrm{dy}\,(x_i - x_{i-1}) - 2\,\mathrm{dx}\,(y_i - y_{i-1}).$$

Известно, что $x_i - x_{i-1} = 1$, тогда

$$d_{i+1} - d_i = 2\,\mathrm{dy} - 2\,\mathrm{dx}\,(y_i - y_{i-1}).$$

Отсюда выразим d_{i+1}:

$$d_{i+1} = d_i + 2\,\mathrm{dy} - 2\,\mathrm{dx}\,(y_i - y_{i-1}).$$

Таким образом, получили итеративную формулу вычисления управляющего коэффициента d_{i+1} по предыдущему значению d_i. С помощью управляющего коэффициента выбирается следующий пиксель – S_i или T_i.

Если $d_i \geq 0$, тогда выбирается T_i и $y_i = y_{i-1} + 1$, $d_{i+1} = d_i + 2\,(dy - dx)$. Если $d_i < 0$, тогда выбирается S_i и $y_i = y_{i-1}$ и $d_{i+1} = d_i + 2\,dy$.

Начальные значения d_1 с учетом того, что $(x_0, y_0) = (0, 0)$,

$$d_1 = 2\,dy - dx.$$

Преимуществом алгоритма является то, что для работы алгоритма требуются минимальные арифметические возможности: сложение, вычитание и сдвиг влево для умножения на 2.

Реализация этого алгоритма выглядит следующим образом:

```
    void MyLine(int x1, int y1, int x2, int y2,
int c)
    {
```



```
int dx, dy, inc1, inc2, d, x, y, Xend;
  dx = abs(x2 - x1);
  dy = abs(y2 - y1);
  d = dy << 1 - dx;
  inc1 = dy << 1;
  inc2 = (dy - dx) << 1;
  if (x1>x2)
     {
     x = x2;
     y = y2;
     Xend = x1;
     }
  else
     {
     x = x1;
     y = y1;
     Xend = x2;
     };
  putpixel(x, y, c);
  while (x < Xend)
    {
    x++;
    if (d < 0) d = d + inc1;
      else
{
y++;
d = d + inc2;
};
    putpixel(x, y, c);
    };
}
```

Если $dy > d$x, то необходимо будет использовать этот же алгоритм, но пошагово увеличивая y и на каждом шаге вычислять x.

10.1.2. Растровая развёртка окружности

Существует несколько очень простых, но не эффективных способов преобразования окружностей в растровую форму. Например, рассмотрим для простоты окружность с центром в начале координат. Ее уравнение записывается как $x^2 + y^2 = R^2$. Решая это уравнение относительно y, получим

$$y = \pm \sqrt{R^2 - x^2}\ .$$

Чтобы изобразить четвертую часть окружности, будем изменять x с единичным шагом от 0 до R и на каждом шаге вычислять y. Вторым простым методом растровой развертки окружности является использование вычислений x и y по формулам $x = R \cos \alpha$, $y = R \sin \alpha$ при пошаговом изменении угла α от 0° до 90°.

Для упрощения алгоритма растровой развёртки стандартной окружности можно воспользоваться её симметрией относительно координатных осей и прямых $y = \pm x$; в случае, когда центр окружности не совпадает с началом координат, эти прямые необходимо сдвинуть параллельно так, чтобы они прошли через центр окружности. Тем самым достаточно построить растровое представление для 1/8 части окружности, а все оставшиеся точки получить симметрией (см. рис. 10.5).

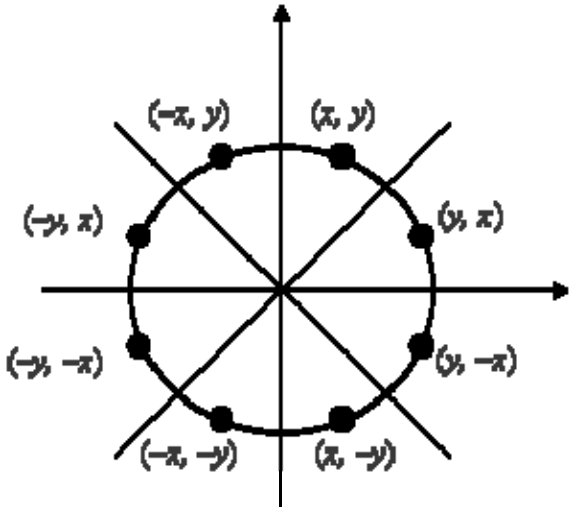


Рис. 10.5. Восьмисторонняя симметрия

Рассмотрим участок окружности из второго октанта $x \in [0, R/\sqrt{2}]$. Далее опишем алгоритм Брезенхейма для этого участка окружности.

На каждом шаге алгоритм выбирает точку P_i (x_i, y_i), которая является ближайшей к истинной окружности. Идея алгоритма

заключается в выборе ближайшей точки при помощи управляющих переменных, значения которых можно вычислить в пошаговом режиме с использованием небольшого числа сложений, вычитаний и сдвигов.

Рассмотрим небольшой участок сетки пикселов, а также возможные способы (от A до E) прохождения истинной окружности через сетку (рис. 10.6).

Предположим, что точка P_{i-1} была выбрана как ближайшая к окружности при $x = x_{i-1}$. Теперь найдем, какая из точек (S_i или T_i) расположена ближе к окружности при $x = x_{i-1} + 1$.

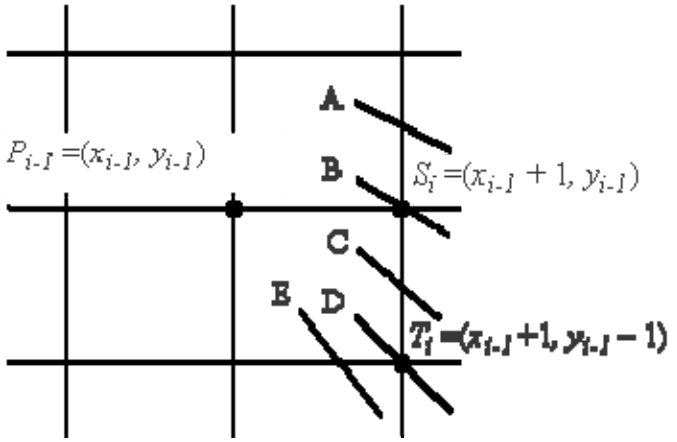


Рис. 10.6. Варианты прохождения окружности через растровую сетку

Заметим, что ошибка при выборе точки P_i (x_i, y_i) была равна

$$D(P_i) = (x_i^2 + y_i^2) - R^2.$$

Запишем выражение для ошибок, получаемых при выборе точки S_i или T_i:

$$D(S_i) = [(x_{i-1} + 1)^2 + (y_{i-1})^2] - R^2;$$

$$D(T_i) = [(x_{i-1} + 1)^2 + (y_{i-1} - 1)^2] - R^2.$$

Если $|\ D(S_i)\ | \geq |\ D(T_i)\ |$, то T_i ближе к реальной окружности, иначе выбирается S_i.

Введем $d_i = | D(S_i) | - | D(T_i) |$.

T_i будет выбираться при $d_i \geq 0$, в противном случае будет устанавливаться S_i.

Опуская алгебраические преобразования, запишем d_i и d_{i+1} для разных вариантов выбора точки S_i или T_i.

$$D_1 = 3 - 2R.$$

Если выбирается S_i (когда $d_i < 0$), то $d_{i+1} = d_i + 4x_{i-1} + 6$.

Если выбирается T_i (когда $d_i \geq 0$), то $d_{i+1} = d_i + 4(x_{i-1} - y_{i-1}) + 10$.

Существует модификация алгоритма Брезенхейма для эллипса.

10.1.3. Закраска области, заданной цветом границы

Рассмотрим область, ограниченную набором пикселей заданного цвета и точку (x, y), лежащую внутри этой области.

Задача заполнения области заданным цветом в случае, когда эта область не является выпуклой, может оказаться довольно сложной.

Простейший рекурсивный алгоритм:

```
     void PixelFill(int x, int y, int border_color,
int color)
     {
     int c = getpixel(x, y);
     if ((c != border_color) && (c != color))
     {
     putpixel(x, y, color);
     PixelFill(x - 1, y, border_color, color);
     PixelFill(x + 1, y, border_color, color);
     PixelFill(x, y - 1, border_color, color);
     PixelFill(x, y + 1, border_color, color);
     }
     }
```

Этот алгоритм является слишком неэффективным, так как для всякого уже отрисованного пикселя функция вызывается ещё 4 раза и, кроме того, этот алгоритм требует слишком большого объёма стека из-

за большой глубины рекурсии. Поэтому для решения задачи закраски области предпочтительнее алгоритмы, способные обрабатывать сразу целые группы пикселей, т. е. использовать их «связность». Если данный пиксель принадлежит области, то, скорее всего, его ближайшие соседи также принадлежат данной области.

Группой таких пикселов обычно выступает полоса, определяемая правым пикселем. Для хранения правых определяющих пикселов используется стек. **Словесно опишем улучшенный алгоритм, использующий когерентность пикселов.**

Сначала заполняется горизонтальная полоса пикселов, содержащих начальную точку. Затем, чтобы найти самый правый пиксель каждой строки, справа налево проверяется строка, предыдущая по отношению к только что заполненной полосе. Адреса найденных пикселов заносятся в стек. То же самое выполняется и для строки, следующей и за последней заполненной полосой. Когда строка обработана таким способом, в качестве новой начальной точки используется пиксель, адрес которого берется из стека. Для него повторяется вся описанная процедура. Алгоритм заканчивает свою работу, если стек пуст.

10.1.4. Заполнение многоугольника

Часто возникает задача заполнения многоугольников, заданных набором вершин.

Задача заполнения многоугольников решается в два этапа:

1) сначала проводится операция отсечения многоугольника;

2) затем производится заполнение полученных многоугольников.

Этап отсечения необходим для определения реальных областей многоугольника, которые будут выведены на экран. Это необходимо, если многоугольник больше или выходит за пределы экрана или окна вывода.

Отсечение многоугольников

Отсечение многоугольников чаще всего проводится для отбрасывания частей многоугольника, выходящих за границу прямоугольной области, которая определяет экран или область окна. Однако отсечение может проводиться относительно и другого многоугольника. При этом порождается новый многоугольник или несколько новых многоугольников.

Рассмотрим алгоритм Сазерленда-Ходгмана (Sutherland-Hodgman). В алгоритме используется стратегия «разделяй и властвуй»,

которая позволяет решение общей задачи свести к решению ряда простых и похожих подзадач. Примером такой подзадачи является отсечение многоугольника относительно одной отсекающей границы. Последовательное решение четырех таких задач позволяет провести отсечение относительно прямоугольной области (рис. 10.7).

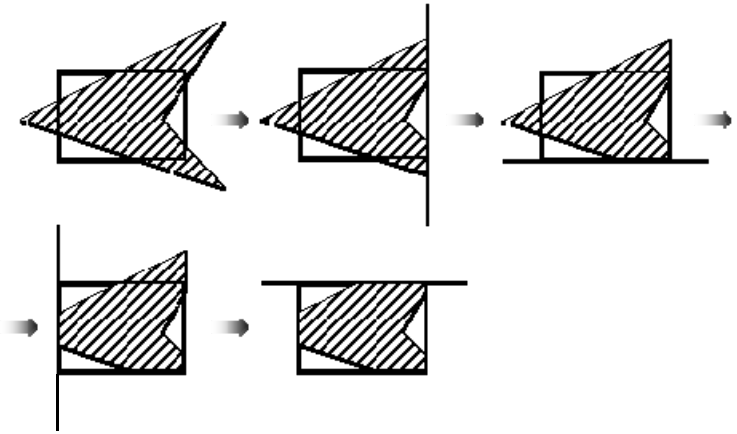


Рис. 10.7. Последовательное отсечение многоугольника

На вход алгоритма поступает последовательность вершин многоугольника V_1, V_2, ..., V_n. Ребра многоугольника проходят от V_i к V_{i+1} от V_n к V_1. С помощью алгоритма производится отсечение относительно ребра и выводится другая последовательность вершин, описывающая усеченный многоугольник.

Алгоритм «обходит» вокруг многоугольника от V_n к V_1 и обратно к V_n, проверяя на каждом шаге соотношение между последовательными вершинами и отсекающей границей. Необходимо проанализировать четыре случая:

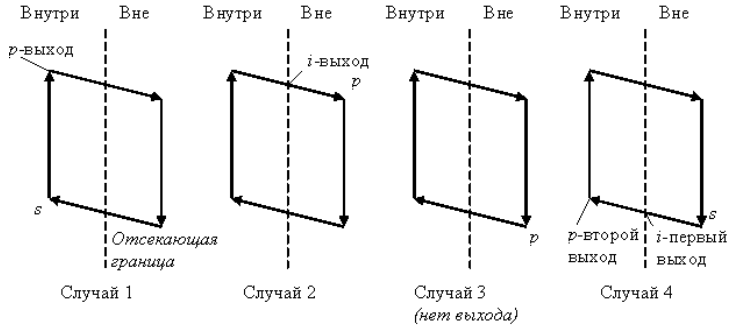


Рис. 10.8. Случаи, возникающие при отсечении многоугольников

Рассмотрим изображенное на рис. 10.8 ребро многоугольника, соединяющее вершину *s* с вершиной *p*. В первом случае ребро полностью лежит внутри отсекающей границы, к выходному списку добавляется вершина *p*. Во втором случае в качестве вершин выводится точка пересечения *i*, поскольку ребро пересекает границу, а начальная точка была выведена при анализе первого случая. В третьем случае обе вершины находятся за пределами границы и ни одна из них не выводится. В четвертом случае к выходному списку добавляется и точка пересечения *i,* и вершина *p*.

При отсечении многоугольника описанным алгоритмом возникает проблема, связанная с тем, что появляются ребра, частично совпадающие с границей окон. Лишние ребра можно устранить, введя дополнительную обработку или воспользовавшись более общим и более сложным алгоритмом Вейлера – Азертона.

Заполнение многоугольников

Рассмотрим, каким образом можно заполнить многоугольник, задаваемый замкнутой ломаной линией без самопересечений.

Простейший способ закраски многоугольника состоит в проверке принадлежности каждой точки этому многоугольнику. Более эффективные алгоритмы используют тот факт, что соседние пикселы, вероятно, имеют одинаковые характеристики (кроме пикселов граничных ребер). Это свойство называется ***пространственной когерентностью***.

В случае с многоугольником когерентность пикселей определяется вдоль сканирующей строки. Сканирующие строки обычно изменяются от «верха» многоугольника до его «низа». Характеристики пикселов изменяются только там, где ребро

многоугольника пересекает строку. Эти пересечения делят сканирующую строку на области закрашенных и не закрашенных пикселов.

Рассмотрим, какие случаи могут возникнуть при делении многоугольника на области сканирующей строкой.

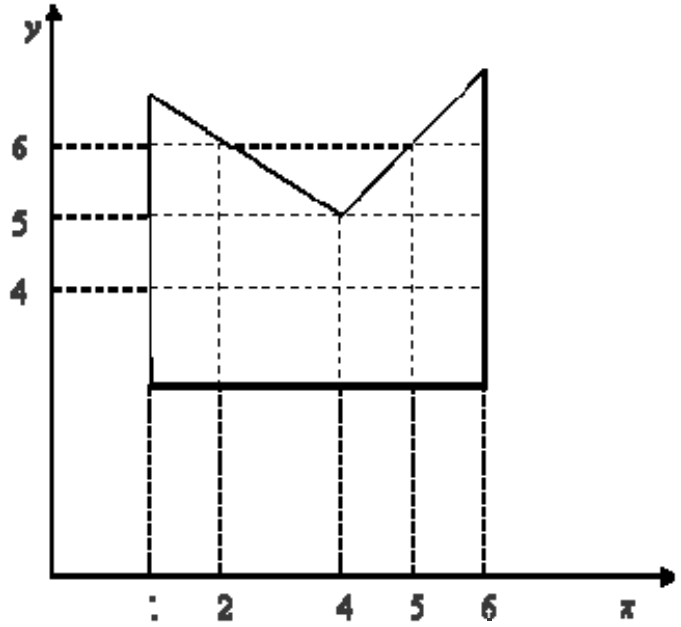


Рис. 10.9. Прохождение сканирующих строк по многоугольнику

1.Простой случай. Например, на рис. 10.9 сканирующая строка $y = 4$ пересекает многоугольник при $x = 1$ и $x = 6$. Получается три области: $x < 1$; $1 \le x \le 6$; $x > 8$. Сканирующая строка $y = 6$ пересекает многоугольник при $x = 1$; $x = 2$; $x = 5$; $x = 6$. Получается пять областей: $x < 1$; $1 \le x \le 2$; $2 < x < 5$; $5 \le x \le 6$; $x > 6$. В этом случае x сортируется в порядке возрастания. Далее список иксов рассматривается попарно. Между парами точек пересечения закрашиваются все пикселы. Для $y = 4$ закрашиваются пикселы в интервале (1, 6), для $y = 6$ закрашиваются пикселы в интервалах (1, 2) и (5, 6).

2.Сканирующая строка проходит через вершину (рис. 10.10).

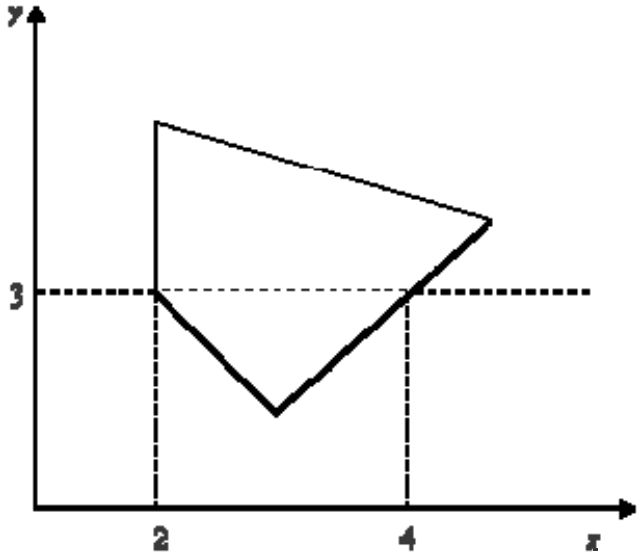


Рис. 10.10. Прохождение сканирующий строки через вершину

Например, по сканирующей строке $y = 3$ упорядоченный список x получится как (2, 2, 4). Вершина многоугольника была учтена дважды, и поэтому закрашиваемый интервал получается неверным: (2, 2). Следовательно, при пересечении вершины сканирующей строкой она должна учитываться единожды. И список по x в приведенном примере будет (2, 4).

3.Сканирующая строка проходит через локальный минимум или максимум (см. рис. 9.9 при $y = 5$). В этом случае учитываются все пересечения вершин сканирующей строкой. На рис. 10.9 при $y = 5$ формируется список x (1, 4, 4, 6). Закрашиваемые интервалы (1, 4) и (4, 6). Условие нахождения локального минимума или максимума определяется при рассмотрении концевых вершин для ребер, соединенных в вершине. Если у обоих концов координаты y больше, чем y вершины пересечения, то вершина – локальный минимум. Если меньше, то вершина пересечения – локальный максимум.

Для ускорения работы алгоритма используется **список активных ребер** (САР). Этот список содержит те ребра многоугольника, которые пересекают сканирующую строку. При пересечении очередной сканирующей строки вершины многоугольника, из САР удаляются ребра, которые находятся выше, и

добавляются концы, которые пересекает сканирующая строка. При работе алгоритма находятся пересечения сканирующей строки только с ребрами из САР.

10.2. Методы устранения ступенчатости

Основная причина появления лестничного эффекта заключается в том, что отрезки, ребра многоугольника, цветовые границы и пр. имеют непрерывную природу, тогда как растровые устройства дискретны.

Лестничный эффект проявляется:

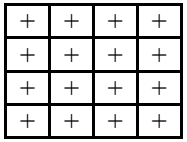
1) при визуализации мелких деталей;
2) при прорисовке ребер и границ;
3) при анимации мелких деталей.

Опишем основные методы устранения ступенчатости.

10.2.1. Метод увеличения частоты выборки

Первый метод устранения ступенчатости связан с увеличением частоты выборки. Увеличение частоты выборки достигается с помощью увеличения разрешения растра. Таким образом учитываются более мелкие детали. Каждый пиксель делится на подпиксели в процессе формирования растра более высокого разрешения. Для получения атрибутов дисплейного пикселя определяются атрибуты в центре каждого подпикселя, которые потом усредняются. Подпиксели в этом случае распределяются равномерно и их атрибуты учитываются одинаково.

Увеличение разрешения в два раза



Увеличение разрешения в четыре раза

В некоторой степени можно получить лучшие результаты, если рассматривать больше подпикселов и учитывать их влияние с помощью весов при определении атрибутов.

1	2	1
2	4	2
1	2	1

1	2	3	4	3	2	1
2	4	6	8	6	4	2
3	6	9	12	9	6	3
4	8	12	16	12	8	4
3	6	9	12	9	6	3
2	4	6	8	6	4	2
1	2	3	4	3	2	1

Числа обозначают относительные веса каждого подпиксела.

10.2.2. Метод, основанный на использовании полутонов

В этом эвристическом методе интенсивность пикселя на ребре устанавливается пропорционально площади части пикселя, находящегося внутри многоугольника.

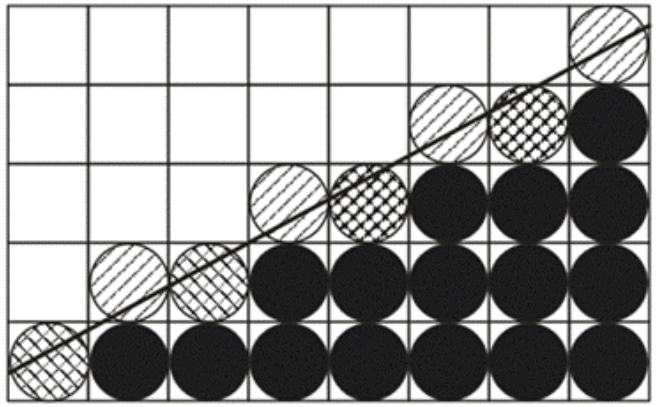


Рис. 10.11. Закраска ребра

На рис. 10.11 приводится многоугольник, ребро которого сгенерировано основным алгоритмом Брезенхейма. Пикселы в этом ребре и внутренние пикселы многоугольника полностью закрашены.

Пикселы, находящиеся выше ребра, закрашиваются с различной интенсивностью, которая пропорциональна площади пиксела, попадающего внутрь многоугольника.

10.3. Простейшие методы обработки изображений

Часто в системах ИИ возникает задача обработки изображений. Обработка, как правило, заключается в наложении на изображение каких-либо эффектов – это размытие, резкость, деформация, шум и т. д., а также в регулировке уровня яркости и контраста.

10.3.1. Яркость и контраст

Яркость и контраст являются субъективными характеристиками изображения, воспринимаемыми человеком.

Яркость представляет собой характеристику, определяющую то, на сколько сильно цвета пикселей отличаются от чёрного цвета. Например, если оцифрованная фотография сделана в солнечную погоду, то ее яркость будет значительной. С другой стороны, если фотография сделана вечером или ночью, то её яркость будет невелика.

Контраст представляет собой характеристику того, насколько большой разброс имеют цвета пикселей изображения. Чем больший разброс имеют значения цветов пикселей, тем больший контраст имеет изображение.

По аналогии с терминами теории вероятностей можно отметить, что яркость представляет собой как бы математическое ожидание значений выборки, а контраст – дисперсию значений выборки.

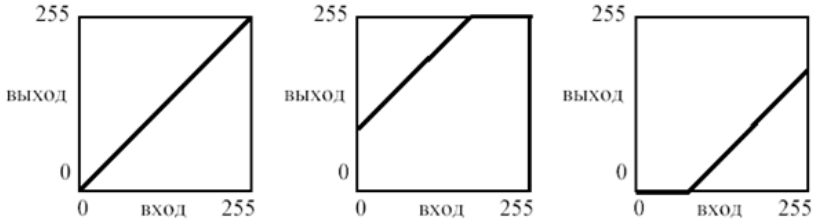
Яркость и контраст могут рассматриваться не только для всего изображения, но и для отдельных фрагментов. Таким образом, возникают понятия **локальной яркости и локального контраста**.

Часто требуется изменить яркость или контраст изображения. Рассмотрим функцию, областью определения и значений которой являются значения цветовых компонент в модели RGB. Аргументом функции является цвет пикселя исходного изображения. Значение функции представляет собой цвет пикселя обработанного изображения. Для изменения яркости/контраста функция применяется для каждого пикселя изображения.

Для нормализации выходных значений функции (они должны принадлежать отрезку [0, 1], как для каждого компонента модели RGB)

используется так называемая арифметика с насыщением. В арифметике с насыщением при возникновении переполнений или заёмов фиксируется наибольшее представимое или наименьшее представимое значения соответственно. Например, если в результате преобразования оказывается, что значение какого-либо компонента модели RGB меньше 0, то берётся значение, равное 0. На практике же **каждый элемент матрицы изображения с 16777216 цветами представляет собой 24-битное значение, где каждый компонент модели RGB представлен 8-ю битами.** Поэтому вместо интервала [0, 1] используется интервал [0, 255].

Если яркость и контраст изображения никак не меняются в процессе преобразования, то функция имеет график, представленный на рис. 10.12, а. Из рисунка видно, что функция в этом случае просто передаёт на выход значение своего аргумента.



а б в

Рис. 10.12. Графики яркости

Яркость для рассматриваемой функции представляет собой сдвиг прямой линии в вертикальном направлении. Яркость изображения увеличивается пропорционально сдвигу прямой. Если прямая сдвигается вверх (рис. 10.12, б), яркость изображения увеличивается, а если прямая сдвигается вниз (рис. 10.12, в) – уменьшается.

Поскольку используется **арифметика с насыщением**, то при установке определённой яркости изображения либо оно полностью окажется засвеченным, либо полностью затемнённым.

При использовании преобразования контраста прямая линия меняет свой наклон. При увеличении контраста изображения (рис. 10.13, а) наклон прямой увеличивается, при уменьшении контраста – уменьшается (рис. 10.13, б). При этом сдвиг прямой в горизонтальном направлении означает, что помимо контраста изменяется и яркость изображения.

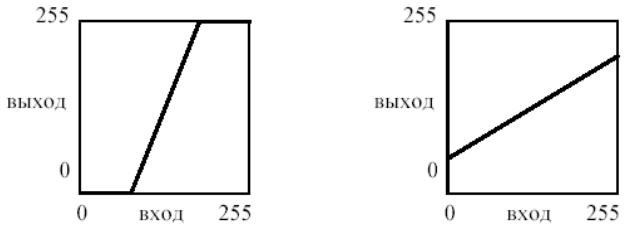


Рис. 10.13. Графики контрастности

Комбинации наклона и сдвига прямой позволяют одновременно изменять и яркость, и контраст изображения. Например, на рис. 10.14 представлен график функции, усиливающей контраст и увеличивающей яркость изображения.

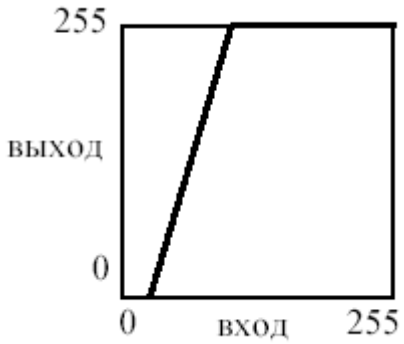


Рис. 10.14. Увеличение яркости и контрастности

Преобразование яркости/контраста может быть применено и к отдельным компонентам модели RGB, например к компоненту красного цвета. Тогда яркость/контраст будут изменяться только для красного компонента, а для других компонент они останутся неизменными. Более того, можно задавать различные преобразования яркости/контраста одновременно для каждого компонента модели RGB.

10.3.2. Масштабирование изображения

Масштабирование изображения позволяет сжать или растянуть его по горизонтали и/или вертикали. При этом изменяется ширина и/или высота изображения. Для масштабирования задаются масштабные коэффициенты – то, насколько нужно сжать/растянуть изображение по горизонтали или вертикали. Масштабные коэффициенты могут задаваться в нормализованной, процентной или непосредственной форме. В нормализованной форме за единицу принимаются размеры исходного изображения. Значения меньше единицы указывают на сжатие изображения, значения больше единицы – на растяжение. В процентной форме нормализованные значения умножаются на 100 %. В непосредственной форме новые размеры по горизонтали и вертикали задаются в виде количества пикселей по тому или другому измерению.

Возникает вопрос о том, каким образом определять цвета при изменении размеров изображения. Существует два основных подхода к этой проблеме:

1. Цвет пикселя в масштабированном изображении принимается равным цвету ближайшего к нему пикселя исходного изображения.

2. Использование интерполяции. В этом случае цвет пикселя масштабируемого изображения вычисляется, как значение некоторой интерполирующей функции от цветов соседних пикселей в исходном изображении.

При использовании билинейной интерполяции цвет вычисляется, как взвешенная сумма ближайших четырёх пикселей исходного изображения (при увеличении) или как взвешенная сумма группы пикселей (при уменьшении).

Первый подход достаточно прост, но не всегда даёт приемлемое качество обработанного изображения. Например, если новый размер намного больше старого, то возникает блочная структура изображения, т. е. каждый пиксель исходного изображения соответствует квадратной области пикселей одного и того же цвета в обработанном изображении. Эта аномалия представлена на рис. 10.15.

С другой стороны, если новый размер намного меньше старого, то при масштабировании одному пикселю обработанного изображения соответствует группа пикселей исходного изображения, причём в процессе масштабирования фактически выбирается случайный пиксель из этой группы.

Рис. 10.15. Некорректное увеличение

Подход, использующий интерполяцию, позволяет достичь более высокого качества изображения, но более сложен в реализации. Обычно используется билинейная или бикубическая интерполяция. Бикубическая интерполяция позволяет получить изображение с более высоким качеством, чем билинейная интерполяция. Однако следует заметить, что при дальнейшем повышении порядка интерполяции качество получаемого изображения может улучшаться незначительно.

Приведем простейшую формулу, которая позволяет определить ближайший пиксель исходного изображения (без использования интерполяции):

$$C_{new}[i][j] = C_{old}[k_1 \cdot i][k_2 \cdot j], \text{ где } i = \overline{0, H_{new}-1},\ j = \overline{0, W_{new}-1};$$

$$k_1 = \frac{H_{old}}{H_{new}},\ k_2 = \frac{W_{old}}{W_{new}}.$$

Параметр W определяет размер изображения по горизонтали, измеряемый в пикселях. Параметр H определяет размер по вертикали. Параметры i и j определяют соответственно строку и столбец матрицы изображения и изменяются в пределах высоты и ширины изображения соответственно.

10.3.3. Преобразование поворота

Преобразование поворота, также как и при рассмотрении плоских геометрических объектов, позволяет поворачивать исходное изображение на заданный угол. Поворот осуществляется вокруг центра изображения. При этом возможны два варианта поворота:

1. Области изображения, вышедшие за его границы при повороте отсекаются, а незаполненные части заполняются каким-либо цветом.

2. Рассчитывается новый размер изображения на основе угла поворота таким образом, чтобы повёрнутое изображение целиком поместилось в новые размеры. Незаполненные части изображения также заполняются каким-либо цветом.

В любом случае для расчёта преобразования поворота может быть использована следующая формула:

$$C_{new}[i][j] = \begin{cases} C_{old}[a][b], a \in [0, H_{old}-1] \wedge b \in [0, W_{old}-1]; \\ C, a \notin [0, H_{old}-1] \vee b \notin [0, W_{old}-1]; \end{cases}$$

$$a = \left| i \cdot \sin(\varphi) + \frac{H_{new}}{2} \right|;\ b = \left| j \cdot \cos(\varphi) + \frac{W_{new}}{2} \right|;$$

$$i = \overline{0, H_{new}-1};\ j = \overline{0, W_{new}-1}.$$

В этой формуле параметр C определяет цвет, которым заполняются пустые участки изображения. Параметр φ определяет угол поворота по часовой стрелке в радианах.

Приведённая формула округляет преобразованные координаты. Однако можно использовать и билинейную интерполяцию, когда цвет пикселя вычисляется как взвешенная сумма цветов четырёх соседних пикселей.

10.3.4. Цифровые фильтры изображений

Цифровые фильтры позволяют накладывать на изображение различные эффекты, например: размытие, резкость, деформацию, шум и т. д.

Цифровой фильтр представляет собой алгоритм обработки изображения. Большая группа цифровых фильтров имеет один и тот

же алгоритм, но эффект, накладываемый фильтром на изображение, зависит от коэффициентов, используемых в алгоритме.

Рассмотрим цифровые фильтры с конечной импульсной характеристикой, основанные на **теории линейных систем и применении двумерных свёрток.**

Свёртка представляет собой способ представления какого-либо векторного значения скалярным значением. Существует бесконечное количество таких способов, многие из которых определяются некоторыми коэффициентами. Применительно к обработке изображений **векторное значение представляет собой цвет группы пикселей, а скалярное значение, получаемое на основе свёртки, представляет собой цвет пикселя,** получаемого в результате применения к исходному изображению какого-либо эффекта.

Цифровые фильтры на основе свёртки характеризуются размером **группы пикселей**. Это называется *размером фильтра*. Также фильтр характеризуется своей **импульсной характеристикой**. Применительно к обработке изображений **импульсная характеристика фильтра представляет собой изображение, получаемое в результате обработки чёрного изображения, в центре которого располагается белая точка.** Конечность импульсной характеристики определяется конечным размером группы пикселей, используемых в фильтре. Импульсная характеристика зависит от размера фильтра и определяется коэффициентами фильтра. **Коэффициенты фильтра представляют собой некоторые скалярные значения, на которые умножаются значения цветов пикселей из группы, соответствующей размеру фильтра.** Обработка изображения с применением такого рода фильтров описывается следующей формулой:

$$C_{new}[i][j] = \sum_{m}\sum_{n} \alpha_{mn} \; C_{old}[i - m/2][j - n/2].$$

В этой формуле коэффициенты α определяют тот эффект, который накладывает фильтр. Константы m и n задают размер фильтра (он является двумерным).

Основной задачей при разработке цифровых фильтров с конечной импульсной характеристикой является расчёт коэффициентов фильтра.

Рассмотрим типовые примеры фильтров с конечной импульсной характеристикой.

Изображение, так же как и звук, может рассматриваться как суперпозиция функций синуса и косинуса с различной амплитудой и фазой. При этом эти **функции являются двумерными**, так как само изображение двумерно. Высокие частоты в изображении означают резкие изменения яркости пикселей. Низкие частоты означают плавные изменения яркости пикселей. На рис. 10.16, а представлено изображение, состоящее из низких частот, а на рис. 10.16, б – из высоких.

Рис. 10.16. Разночастотные изображения

Часто возникает задача увеличения резкости изображения, что означает усиление высоких частот. Также может возникать задача уменьшения резкости и увеличения размытости, что означает усиление низких частот.

Следующая матрица определяет коэффициенты цифрового фильтра размером 3 на 3 пикселя, используемого для повышения резкости изображения:

$-k/8$	$-k/8$	$-k/8$
$-k/8$	$k+1$	$-k/8$
$-k/8$	$-k/8$	$-k/8$

,

где параметр k определяет степень повышения контраста. Обычно используется $k = 2$. Следующая матрица определяет коэффициенты цифрового фильтра размером 3 на 3 пикселя, используемого для повышения размытости изображения:

1/9	1/9	1/9
1/9	1/9	1/9
1/9	1/9	1/9

.

В данной матрице значения всех коэффициентов одинаковы. Это практически означает, что цвета группы пикселей 3 на 3 просто усредняются. Размер матрицы определяет степень размытости. Например, если используется матрица 5 на 5, где каждый элемент равен 1/25, то размытость изображения будет больше, чем при использовании матрицы 3 на 3. Также часто используется размытие по Гауссу.

Из приведённых выше примеров видно, что сумма всех коэффициентов равна единице. Это основное условие, используемое при расчёте фильтров с конечной импульсной характеристикой. Несоблюдение этого условия приводит к тому, что контраст изображения после применения фильтра изменяется.

Также отметим, что размер фильтра всегда является нечётным (например, 3 на 3 или 5 на 5).

11. Компьютерная геометрия в искусственном интеллекте

11.1. Двумерные преобразования

Компьютерная геометрия в искусственном интеллекте есть математический аппарат, положенный в основу графики систем искусственного интеллекта. В свою очередь, основу **компьютерной геометрии составляют различные преобразования точек и линий.** При использовании машинной графики системах искусственного интеллекта можно по желанию изменять масштаб изображения, вращать его, смещать и трансформировать для улучшения наглядности перспективного изображения. **Все эти преобразования можно выполнить на основе математических методов**, которые мы будем рассматривать далее.

Преобразования, как и компьютерную геометрию, разделяют на двумерные (или преобразования на плоскости) и трехмерные (или пространственные). Вначале рассмотрим преобразования на плоскости.

Для начала заметим, что точки на плоскости задаются с помощью двух ее координат. Таким образом, геометрически каждая точка задается значениями координат вектора относительно выбранной системы координат. **Координаты точек можно рассматривать как элементы матрицы [*x,y*], т. е. в виде вектор-строки или вектор-столбца. Положением этих точек управляют путем преобразования матрицы**.

Точки на плоскости *xy* можно перенести в новые позиции путем добавления к координатам этих точек констант переноса:

$$\begin{bmatrix} x' & y' \end{bmatrix} = \begin{bmatrix} x & y \end{bmatrix} + \begin{bmatrix} a & b \end{bmatrix} = \begin{bmatrix} x+a & y+b \end{bmatrix}.$$

Таким образом, для перемещения точки на плоскости надо к матрице ее координат прибавить матрицу коэффициентов преобразования.

Рассмотрим результаты матричного умножения матрицы $[x,y]$, определяющей точку P, и матрицы преобразований 2×2 общего вида:

$$[x \quad y]\cdot\begin{bmatrix} a & b \\ c & d \end{bmatrix} = [(ax+cy)(bx+dy)] = [x^* \quad y^*]$$

Проведем анализ полученных результатов, рассматривая x^* и y^* как преобразованные координаты. Для этого исследуем несколько частных случаев.

Рассмотрим случай, когда $a = d = 1$ и $c = b = 0$. Матрица преобразований приводит к матрице, идентичной исходной:

$$[x \quad y]\begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix} = [(1x+0y)(0x+1y)] = [x \quad y] = [x^* \quad y^*]$$

При этом изменений координат точки P не происходит.

Если теперь $d = 1$, $b = c = 0$, a = const, то:

$$[x \quad y]\begin{bmatrix} a & 0 \\ 0 & 1 \end{bmatrix} = [(ax+0y)(0x+1y)] = [ax \quad y] = [x^* \quad y^*]$$

Как видно, это приводит к изменению масштаба в направлении x, так как $x^*=ax$. Следовательно, данное матричное преобразование эквивалентно перемещению исходной точки в направлении x.

Теперь положим $b = c = 0$, т. е.

$$[x \quad y]\begin{bmatrix} a & 0 \\ 0 & d \end{bmatrix} = [(ax+0y)(0x+dy)] = [ax \quad dy] = [x^* \quad y^*]$$

В результате получаем изменение масштабов в направлениях x и y. Если $a \neq d$, то перемещения вдоль осей неодинаковы. Если $a = d > 1$, то имеет место увеличение масштаба координат точки P. Если $0 < a=d < 1$, то будет иметь место уменьшение масштаба координат точки P.

Если a или (и) d отрицательны, то происходит отображение координат точек. Рассмотрим это, положив $b = c = 0$; $d = 1$ и $a = -1$, тогда

$$[x \quad y]\cdot\begin{bmatrix} -1 & 0 \\ 0 & 1 \end{bmatrix} = [(-1x+0y)(0x+1y)] = [-x \quad y] = [x^* \quad y^*]$$

Произошло отображение точки относительно оси y. В случае $b = c = 0$, $a = 1$, $d = -1$, отображение происходит относительно оси x. Если $b = c = 0$, $a = d < 0$, то отображение будет происходить относительно начала координат.

Заметим, что отображение и изменение масштаба вызывают только диагональные элементы матрицы преобразования.

Теперь рассмотрим случай, когда $a = d = 1$, а $c = 0$, т. е.

$$[x \quad y]\cdot\begin{bmatrix} 1 & b \\ 0 & 1 \end{bmatrix} = [x \quad (bx+y)] = [x^* \quad y^*]$$

Координата x точки P не изменяется, в то время как y^* линейно зависит от начальных координат. Этот эффект называется *сдвигом*. Аналогично, когда $a = d = 1$, $b = 0$, преобразование осуществляет сдвиг пропорционально координате y.

Заметим, что преобразование общего вида, примененное к началу координат, не приведет к изменению координат точки (0,0). Следовательно, начало координат инвариантно при общем преобразовании. Это ограничение преодолевается за счет использования однородных координат.

Если подвергнуть общему преобразованию различные геометрические фигуры, то можно установить, что параллельные прямые

преобразуются в параллельные прямые, середина отрезка – в середину отрезка, параллелограмм – в параллелограмм, точка пересечения двух линий – в точку пересечения преобразованной пары линий.

Преобразование единичного квадрата

Четыре вектора положения точек единичного квадрата с одним углом в начале координат записываются в виде

$$\begin{bmatrix} 0 & 0 \\ 1 & 0 \\ 1 & 1 \\ 0 & 1 \end{bmatrix} \begin{matrix} \rightarrow A \\ \rightarrow B \\ \rightarrow C \\ \rightarrow D \end{matrix}$$

Применение общего матричного преобразования

$$\begin{bmatrix} a & b \\ c & d \end{bmatrix}$$

к единичному квадрату приводит к следующему:

$$\begin{matrix} A \rightarrow \\ B \rightarrow \\ C \rightarrow \\ D \rightarrow \end{matrix} \begin{bmatrix} 0 & 0 \\ \vdots & 0 \\ \vdots & 1 \\ 0 & 1 \end{bmatrix} \cdot \begin{bmatrix} a & b \\ c & d \end{bmatrix} = \begin{bmatrix} 0 & 0 \\ a & b \\ a+c & b+d \\ c & d \end{bmatrix} \begin{matrix} \leftarrow A^* \\ \leftarrow B^* \\ \leftarrow C^* \\ \leftarrow D^* \end{matrix}$$

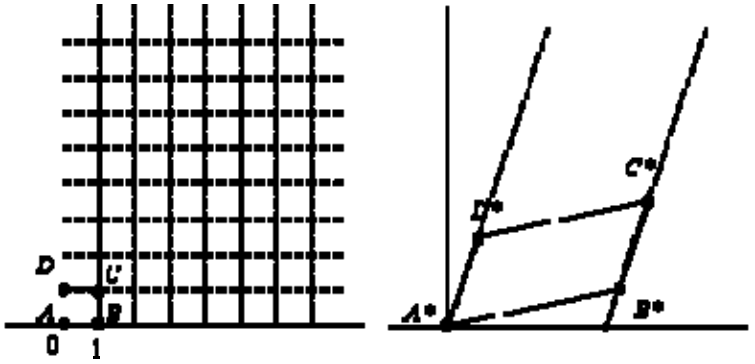


Рис. 11.1. Преобразования единичного квадрата

Из полученного соотношения можно сделать вывод, что координаты B^* определяются первой строкой матрицы преобразования, а координаты D^* второй строкой этой матрицы. Таким образом, если координаты точек B^* и D^* известны, то общая матрица преобразования определена. Воспользуемся этим свойством для нахождения матрицы преобразования для вращения на произвольный угол. Общую матрицу 2×2, которая осуществляет вращение фигуры относительно начала координат, можно получить из рассмотрения вращения единичного квадрата вокруг начала координат.

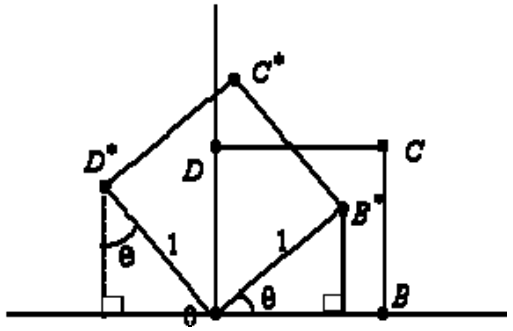


Рис. 11.2. Вращение единичного квадрата

Как следует из рис. 11.2, точка B с координатами (1,0) преобразуется в точку B^*, для которой $x^*=(1)\cos\theta$ и $y=(1)\sin\theta$, а точка D, имеющая координаты (0,1) переходит в точку D^* с координатами $x^*=(-1)\sin\theta$ и $y^*=(1)\cos\theta$.

Матрица преобразования общего вида записывается так:

$$\begin{bmatrix} \cos\theta & \sin\theta \\ -\sin\theta & \cos\theta \end{bmatrix}.$$

Для частных случаев. Поворот на 90° можно осуществить с помощью матрицы преобразования

$$\begin{bmatrix} 0 & 1 \\ -1 & 0 \end{bmatrix}.$$

Если использовать матрицу координат вершин, то получим, например:

$$\begin{bmatrix} 3 & -1 \\ 4 & 1 \\ 2 & 1 \end{bmatrix} \cdot \begin{bmatrix} 0 & 1 \\ -1 & 0 \end{bmatrix} = \begin{bmatrix} 1 & 3 \\ -1 & 4 \\ -1 & 2 \end{bmatrix}.$$

Поворот на 180° получается с помощью матрицы

$$\begin{bmatrix} -1 & 0 \\ 0 & -1 \end{bmatrix}.$$

Отображение

В то время как чистое двумерное вращение в плоскости *xy* осуществляется вокруг оси, перпендикулярной к этой плоскости, отображение определяется поворотом на 180° вокруг оси, лежащей в плоскости *xy*.

Такое вращение вокруг линии $y = x$ происходит при использовании матрицы

$$\begin{bmatrix} 0 & 1 \\ 1 & 0 \end{bmatrix}.$$

Преобразованные новые выражения определяются соотношением

$$\begin{bmatrix} 0 & 1 \\ 7 & 3 \\ 6 & 2 \end{bmatrix} \cdot \begin{bmatrix} 0 & 1 \\ 1 & 0 \end{bmatrix} = \begin{bmatrix} 1 & 0 \\ 3 & 7 \\ 2 & 6 \end{bmatrix}.$$

Вращение вокруг $y = 0$ получается при использовании матрицы

$$\begin{bmatrix} 1 & 0 \\ 0 & -1 \end{bmatrix}.$$

11.1.1. Однородные координаты

Преобразования переноса, масштабирования и поворота записываются в матричной форме в виде

$$P^* = P + T,$$

$$P^* = P \cdot S,$$

$$P^* = P \cdot R.$$

Очевидно, что перенос, в отличие от масштабирования и поворота, реализуется с помощью сложения. Это обусловлено тем, что вводить константы переноса внутрь структуры общей матрицы размером 2×2 не представляется возможным. Желательным является представление преобразований в единой форме – с помощью умножения матриц. Эту проблему можно решить за счет введения третьей компоненты в векторы точек $[x \quad y]$ и $[x^* \quad y^*]$, т. е. представляя их в виде $[x \quad y \quad 1]$ и $[x^* \quad y^* \quad 1]$. Матрица преобразования после этого становится матрицей размером 3×2:

$$\begin{bmatrix} 1 & 0 \\ 0 & 1 \\ m & n \end{bmatrix}.$$

Это необходимо, поскольку число столбцов в матрице, описывающей точку, должно равняться числу строк в матрице преобразования для выполнения операции умножения матриц. Таким образом,

$$[x \quad y \quad 1]\cdot\begin{bmatrix}1 & 0\\ 0 & 1\\ m & n\end{bmatrix}=[x+m \quad y+n]=[x^* \quad y^*]$$

,

откуда следует, что константы *m, n* вызывают смещение *x** и *y** относительно *x* и *y*. Поскольку матрица 3×2 не является квадратной, она не имеет обратной матрицы. Эту трудность можно обойти, дополнив матрицу преобразования до квадратной размером 3×3. Например,

$$\begin{bmatrix}1 & 0 & 0\\ 0 & 1 & 0\\ m & n & 1\end{bmatrix}.$$

Заметим, что третья компонента векторов положения точек не изменяется при добавлении третьего столбца к матрице преобразования. Используя эту матрицу в соотношении, получаем преобразованный вектор [*x* y** 1]. Добавление третьего элемента к вектору положения и третьего столбца к матрице преобразования позволяет выполнить смещение вектора положения. Третий элемент здесь можно рассматривать как дополнительную координату вектора положения. Итак, вектор положения [*x y* 1] при воздействии на него матрицы 3×3 становится вектором положения в общем случае вида [*X Y H*]. Представленное преобразование было выполнено так, что

[X *Y H*] = [*x* y** 1].

Преобразование, имеющее место в трехмерном пространстве, в нашем случае ограничено плоскостью, поскольку *H* = 1. Если, однако, третий столбец

$$\begin{bmatrix}p\\ q\\ s\end{bmatrix}$$

матрицы преобразования *T* размера 3 x 3 отличен от 0, то в результате матричного преобразования получим [*x y* 1] =*[X Y H],* где $H \neq 1$.

Плоскость, в которой теперь лежит преобразованный вектор положения, находится в трехмерном пространстве. Однако сейчас нас не интересует то, что происходит в трехмерном пространстве.

Итак, найденные *x** и *y** получены с помощью пучка лучей, проходящих через начало координат. Результат преобразований показан на рис. 11.3.

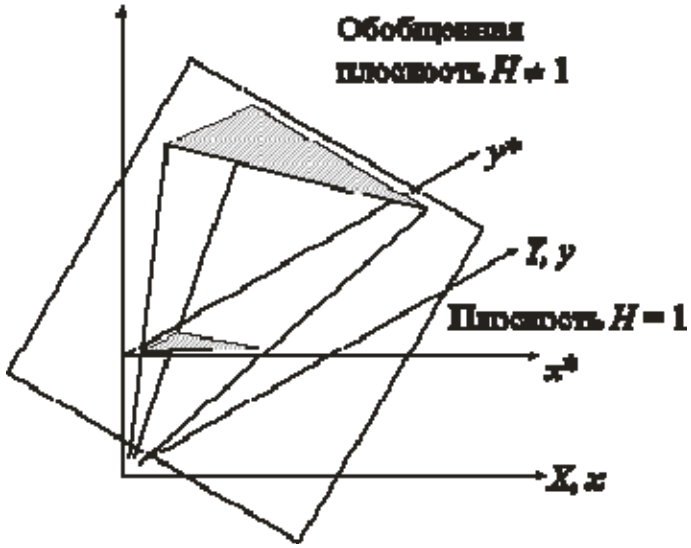


Рис. 11.3. Геометрическое представление однородных координат

Из рассмотрения подобных треугольников видно, что

$$H/X = 1/x^* \text{ и } H/Y = 1/y^*.$$

Рассматривая три компоненты, запишем это в виде

$$[x^* \quad y^* \quad 1] = \left[\frac{X}{H} \quad \frac{Y}{H} \quad 1\right].$$

Представление двумерного вектора трехмерным или в общем случае n-мерного вектора $(n + 1)$-мерным называют *однородным координатным воспроизведением*. При однородном координатном воспроизведении n-мерного вектора оно выполняется в $(n + 1)$-мерном пространстве, и конечные результаты в n-мерном пространстве получают с помощью обратного преобразования. Таким образом, двумерный вектор $[x\ y]$ представляется трехкомпонентным вектором $[hx \quad hy \quad h]$. Разделив компоненты вектора на однородную координату h, получим

$$x = \frac{hx}{h} \text{ и } y = \frac{hy}{h}.$$

Не существует единственного однородного координатного представления точки в двумерном пространстве. Например, однородные координаты (12, 8, 4), (6, 4, 2) и (3, 2, 1) представляют исходную точку [3 2]. Для простоты вычислений выбираем $[x\ y\ 1]$, чтобы представить непреобразованную точку в двумерных однородных координатах. Преобразование

$$[x^* \quad y^*] = [x \quad y]\begin{bmatrix} a & b \\ c & d \end{bmatrix}$$

в дополнительных координатах задается выражением в однородных координатах в виде

$$[X \quad Y \quad H] = [x \quad y \quad 1]\cdot\begin{bmatrix} a & b & 0 \\ c & d & 0 \\ 0 & 0 & 1 \end{bmatrix}.$$

Выполнение указанных выше преобразований показывает, что $X = x^*$, $Y = y^*$, а $H = 1$. Равенство единице дополнительной координаты означает, что преобразованные однородные координаты равны исходным координатам.

В общем случае $H \neq 1$, и преобразованные обычные координаты получаются за счет нормализации однородных координат, т. е.

$$x^* = \frac{X}{H} \text{ и } y^* = \frac{Y}{H}.$$

Геометрически все преобразования x и y происходят в плоскости $H = 1$ после нормализации преобразованных однородных координат.

Преимущество введения однородных координат проявляется при использовании матрицы преобразований общего вида порядка 3×3

$$\begin{bmatrix} a & b & p \\ c & d & q \\ m & n & s \end{bmatrix},$$

с помощью которой можно выполнять и другие преобразования, такие как смещение, операции изменения масштаба и сдвига, обусловленные матричными элементами a, b, c и d. Указанные операции рассмотрены ранее.

Чтобы показать воздействие третьего столбца матрицы преобразований 3×3, рассмотрим следующую операцию:

$$[X \quad Y \quad H] = [x \quad y \quad 1] \cdot \begin{bmatrix} 1 & 0 & p \\ 0 & 1 & q \\ 0 & 0 & 1 \end{bmatrix} = [x \quad y \quad (px + qy + 1)]$$

;

здесь $X = x$, $Y = y$, а $H = px + qy + 1$. Переменная H, которая определяет плоскость, содержащую преобразованные точки, представленные в однородных координатах, теперь образует уравнение плоскости в трехмерном пространстве.

Это преобразование показано на рис. 11.4, где линия AB, лежащая в плоскости xy, спроектирована на линию CD плоскости $pX + qY - H + 1 = 0$.

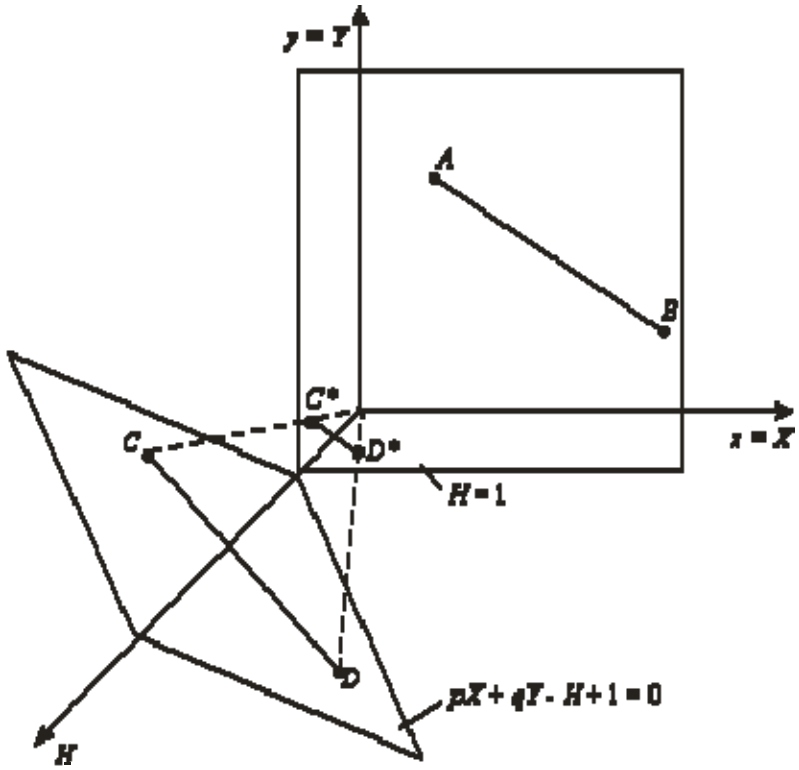


Рис. 11.4. Преобразование линии в однородных координатах

На рисунке величина $p = q = 1$.Выполним нормализацию для того, чтобы получить обычные координаты:

$$x^* = \frac{X}{H} = \frac{X}{pX + qY - 1},$$

$$y^* = \frac{Y}{H} = \frac{Y}{pX + qY + 1}$$

Полагая $p = q = 1$, для изображенных на рисунке точек A и B с координатами соответственно (1, 3) и (4, 1) получим

$$x^* = \frac{1}{:+3+1} = \frac{1}{5} \text{ и } y^* = \frac{3}{5}.$$

После преобразования A в C^* и B в D^* имеем

$$x^* = \frac{4}{1+4+1} = \frac{2}{3} \text{ и } y^* = \frac{1}{6}.$$

Однородные координаты для точек C^* и D^*, показанные на рисунке, соответственно равны

$$\left(\frac{1}{5}, \frac{3}{5}, 1\right) \text{ и } \left(\frac{2}{3}, \frac{1}{6}, 1\right).$$

Результатом нормализации является перевод трехмерной линии CD в ее проекцию C^*D^* на плоскость $H = 1$. Как показано на рисунке, центром проекции является начало координат.

Основная матрица преобразования размером 3×3 для двумерных однородных координат может быть подразделена на четыре части:

$$\left[\begin{array}{cc|c} a & b & p \\ c & d & q \\ \hline m & n & s \end{array}\right].$$

Как мы видим, a, b, c и d осуществляют изменение масштаба, сдвиг и вращение; m и n выполняют смещение, а p и q — получение проекций. Оставшаяся часть матрицы, элемент s, производит полное изменение масштаба. Чтобы показать это, рассмотрим преобразование

$$[X \; Y \; H] = [x \; y \; 1] \cdot \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & s \end{bmatrix} = [x \; y \; s]$$

Здесь $X = x$, $Y = y$, а $H = s$. Это дает $x^* = x/s$ и $y^* == y/s$. В результате преобразования $[x\ y\ 1\ ] \longrightarrow [x/s\ y/s\ 1]$ имеет место однородное изменение масштаба вектора положения. При $s < 1$ происходит увеличение, а при $s > 1$ — уменьшение масштаба.

11.1.2. Двумерное вращение вокруг произвольной оси

Выше было рассмотрено вращение изображения около начала координат. Однородные координаты обеспечивают поворот изображения вокруг точек, отличных от начала координат. В общем случае вращение около произвольной точки может быть выполнено путем переноса центра вращения в начало координат, поворотом относительно начала координат, а затем переносом точки вращения в исходное положение. Таким образом, поворот вектора положения $[x\ y\ 1\ ]$ около точки (m, n) на произвольный угол может быть выполнен с помощью преобразования

$$[x \; y \; 1] \cdot \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ -m & -n & 1 \end{bmatrix} \cdot \begin{bmatrix} \cos\theta & \sin\theta & 0 \\ -\sin\theta & \cos\theta & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ m & n & 1 \end{bmatrix} = [X \; Y \; H].$$

Выполнив две операции умножения матриц, можно записать

$$[X \; Y \; H] = [x \; y \; 1] \cdot \begin{bmatrix} \cos\theta & \sin\theta & 0 \\ -\sin\theta & \cos\theta & 0 \\ -m(\cos\theta - 1) + n(\sin\theta) & -m(\sin\theta) - n(\cos\theta - 1) & 1 \end{bmatrix}$$

Предположим, что центр изображения имеет координаты (4, 3) и желательно повернуть изображение на 90° против часовой стрелки вокруг центральной его оси. Действие, выполненное с помощью матрицы

$$\begin{bmatrix} c & 1 & 0 \\ 1 & 0 & 0 \\ c & 0 & 1 \end{bmatrix},$$

вызывает вращение вокруг начала координат, а не вокруг оси. Как сказано выше, необходимо вначале осуществить перенос изображения таким образом, чтобы желаемый центр вращения находился в начале координат. Это осуществляется с помощью матрицы переноса

$$\begin{bmatrix} : & 0 & 0 \\ 0 & 1 & 0 \\ -4 & -3 & 1 \end{bmatrix}.$$

Затем следует применить матрицу вращения и, наконец, привести результаты к началу координат посредством обратной матрицы. Вся операция

$$[X \quad Y \quad H] = [x \quad y \quad 1] \cdot \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ -4 & -3 & 1 \end{bmatrix} \cdot \begin{bmatrix} 0 & 1 & 0 \\ -1 & 0 & 0 \\ 0 & 0 & 1 \end{bmatrix} \cdot \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 4 & 3 & 1 \end{bmatrix}$$

может быть объединена в одну матричную операцию путем выполнения матричных преобразований вида

$$[X \quad Y \quad H] = [x \quad y \quad 1] \cdot \begin{bmatrix} 0 & 1 & 0 \\ -1 & 0 & 0 \\ 7 & -1 & 1 \end{bmatrix}.$$

В результате будет получено $x^* = X/H$ и $y^* = Y/H$. Двумерные вращения около каждой оси ортогональной системы представлены на рис. 11.5.

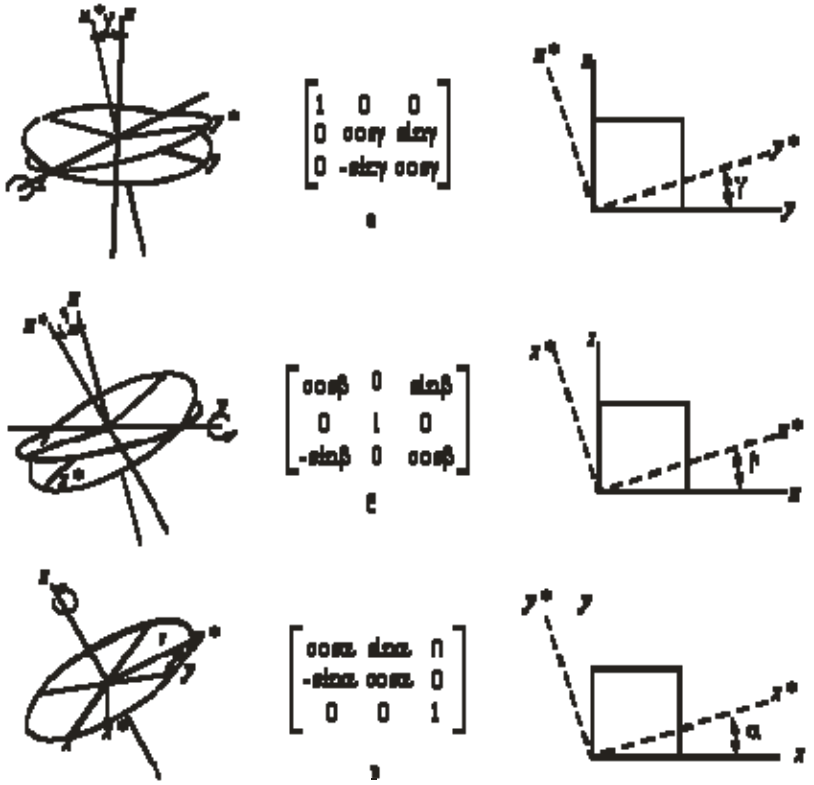


Рис. 11.5. Вращение: а — вокруг оси *x;* б — вокруг оси y; в — вокруг оси z

11.2. Трехмерные преобразования и проекции

Рассмотрим трехмерную декартовую систему координат, являющуюся *правосторонней*. Примем соглашение, в соответствии с которым будем считать положительными такие повороты, при которых (если смотреть с конца полуоси в направлении начала координат) поворот на 90° против часовой стрелки будет переводить одну полуось в другую. На

основе этого соглашения строится следующая таблица, которую можно использовать как для правых, так и для левых систем координат:

Если ось вращения	Положительным будет направление поворота
X	От *y* к *z*
Y	От *z* к *x*
Z	От *x* к *y*

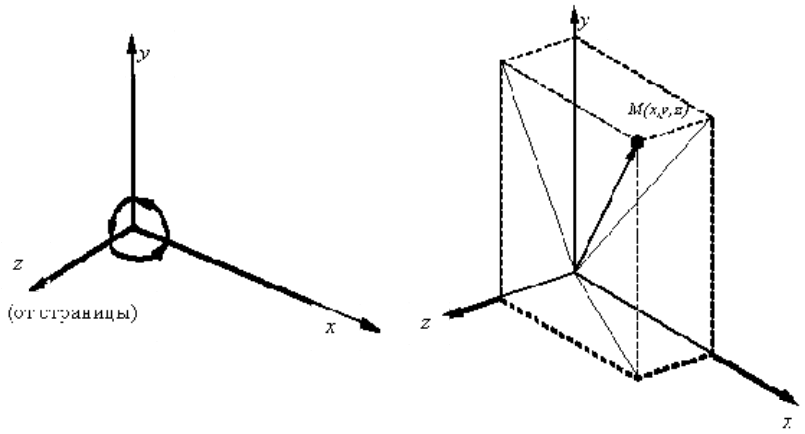


Рис. 11.6. Трехмерная система координат

Аналогично тому, как точка на плоскости описывается вектором (*x,y*), точка в трехмерном пространстве описывается вектором (*x,y,z*).

Как и в двухмерном случае, для возможности реализаций трехмерных преобразований с помощью матриц перейдем к однородным координатам:

[*x,y,x*,1] или [*X,Y,Z,H*]

$[x^*,y^*,z^*1] = [\ \frac{X}{H}, \frac{Y}{H}, \frac{Z}{H}, 1\ ]$, где $H \neq 1$, $H \neq 0$.

Обобщенная матрица преобразования 4×4 для трехмерных однородных координат имеет вид

$$T = \begin{bmatrix} a & b & c & p \\ d & e & f & q \\ h & i & j & r \\ l & m & n & s \end{bmatrix}$$

Эта матрица может быть представлена в виде четырех отдельных частей:

$$\begin{bmatrix} 3\times3 & 3\times1 \\ 1\times3 & 1\times1 \end{bmatrix}$$

- Матрица 3×3 осуществляет линейное преобразование в виде изменения масштаба, сдвига и вращения. (***Линейное преобразование*** трансформирует исходную линейную комбинацию векторов в некоторую линейную их комбинацию.)

- Матрица 1×3 производит перенос.

- Матрица 3×1- преобразования в перспективе.
- Скалярный элемент 1×1 выполняет общее изменение масштаба.

Рассмотрим воздействие матрицы 4×4 на однородный вектор [*x,y,z,*1]:

1. Трехмерный перенос – является простым расширением двумерного:

$$T(Dx,Dy,Dz) = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ Dx & Dy & Dz & 1 \end{bmatrix}$$

т. е. $[x,y,z,1]*T(Dx,Dy,Dz)=[x+Dx,y+Dy,z+Dz,1]$.

2. Трехмерное изменение масштаба.

Рассмотрим **частичное** изменение масштаба. Оно реализуется следующим образом:

$$S(Sx,Sy,Sz)=\begin{bmatrix} Sx & 0 & 0 & 0 \\ 0 & Sy & 0 & 0 \\ 0 & 0 & Sz & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

т. е. [x,y,z,1]*S(Sx,Sy,Sz)=[Sx*x,Sy*y,Sz*z,1].

Общее изменение масштаба получается за счет 4-го диагонального элемента, т. е.

$$[x\ y\ z\ 1] * \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & S \end{bmatrix} = [x\ y\ z\ S] = [x^*\ y^*\ z^*\ \ 1] = [\ \frac{X}{S}, \frac{Y}{S}, \frac{Z}{S}, 1\ ].$$

Такой же результат можно получить при равных коэффициентах частичных изменений масштабов. В этом случае матрица преобразования такова:

$$S=\begin{bmatrix} \frac{1}{S} & 0 & 0 & 0 \\ 0 & \frac{1}{S} & 0 & 0 \\ 0 & 0 & \frac{1}{S} & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

3. Трехмерный сдвиг

Недиагональные элементы матрицы 3×3 осуществляют сдвиг в трех измерениях, т. е.

$$[x\ y\ z\ 1] * \begin{bmatrix} 1 & b & c & 0 \\ d & 1 & f & 0 \\ h & i & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} = [x+yd+hz,\ bx+y+iz,\ cx+fy+z,\ 1].$$

4. Трехмерное вращение

Двухмерный поворот, рассмотренный ранее, является в то же время трехмерным поворотом вокруг оси *Z* . В трехмерном пространстве поворот вокруг оси *Z* описывается матрицей

$$R_z(\Theta)=\begin{bmatrix} \cos(\Theta) & \sin(\Theta) & 0 & 0 \\ -\sin(\Theta) & \cos(\Theta) & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

Матрица поворота вокруг оси *X* имеет вид

$$R_x(\Theta)=\begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & \cos(\Theta) & \sin(\Theta) & 0 \\ 0 & -\sin(\Theta) & \cos(\Theta) & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

Матрица поворота вокруг оси *Y* имеет вид

$$R_y(\Theta)=\begin{bmatrix}\cos(\Theta) & 0 & -\sin(\Theta) & 0\\ 0 & 1 & 0 & 0\\ \sin(\Theta) & 0 & \cos(\Theta) & 0\\ 0 & 0 & 0 & 1\end{bmatrix}$$

Результатом произвольной последовательности поворотов вокруг осей *x, y, z* является матрица

$$A=\begin{bmatrix}a & b & c & 0\\ d & e & f & 0\\ h & i & j & 0\\ 0 & 0 & 0 & 1\end{bmatrix}$$

Подматрицу 3×3 называют ортогональной, так как ее столбцы являются взаимно ортогональными единичными векторами.

Матрицы поворота сохраняют длину и углы, а матрицы масштабирования и сдвига нет.

11.3. Проекции

В общем случае проекции преобразуют точки, заданные в системе координат размерностью *n*, в системы координат размерностью меньше чем *n*.

Будем рассматривать случай проецирования трех измерений в два. Проекция трехмерного объекта (представленного в виде совокупности точек) строится при помощи прямых проекционных лучей, которые называются ***проекторами*** и которые проходят через каждую точку объекта и, пересекая картинную плоскость, образуют ***проекцию***.

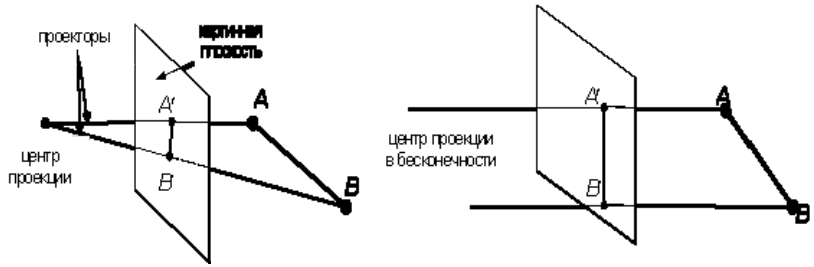


Рис. 11.7. Центральная и параллельная проекции

Определенный таким образом класс проекций существует под названием ***плоских геометрических проекций***, так как проецирование производится на плоскость, а не на искривленную поверхность и в качестве проекторов используются прямые, а не кривые линии.

Многие картографические проекции являются либо не плоскими, либо не геометрическими.

Плоские геометрические проекции в дальнейшем будем называть просто проекциями.

Проекции делятся на два основных класса (рис. 11.7):

- параллельные (аксонометрические);
- центральные (перспективные).

Полная классификация проекций приведена на рис. 10.8.

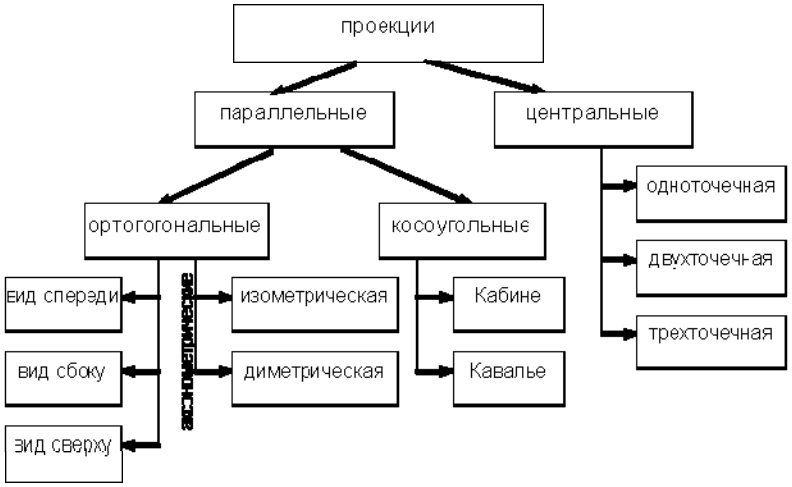


Рис. 11.8. Классификация проекций

Параллельные проекции делятся на два типа в зависимости от соотношения между направлением проецирования и нормалью к проекционной плоскости (рис. 11.9.):

1) ***ортографические*** – направления совпадают, т. е. направление проецирования является нормалью к проекционной плоскости;

2) ***косоугольные*** – направление проецирования и нормаль к проекционной плоскости не совпадают.

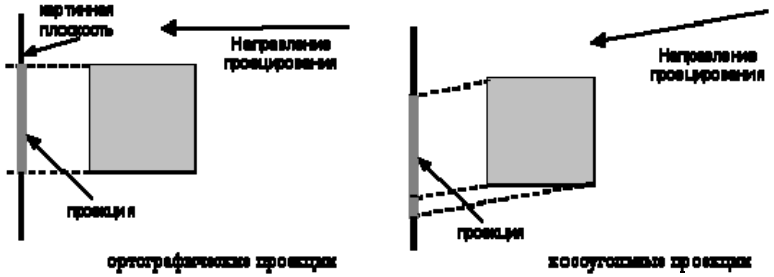


Рис. 11.9. Ортографические и косоугольные проекции

Наиболее широко используемыми видами ортографических проекций является вид спереди, вид сверху(план) и вид сбоку, в которых картинная плоскость перпендикулярна главным координатным осям. Если проекционные плоскости не перпендикулярны главным координатным осям, то такие проекции называются ***аксонометрическими***.

При аксонометрическом проецировании сохраняется параллельность прямых, а углы изменяются; расстояние можно измерить вдоль каждой из главных координатных осей (в общем случае с различными масштабными коэффициентами).

Изометрическая проекция – нормаль к проекционной плоскости, (а следовательно и направление проецирования) составляет равные углы с каждой из главных координатных осей. Если нормаль к проекционной плоскости имеет координаты (*a,b,c*), то потребуем, чтобы $|a| = |b| = |c|$, или $\pm a = \pm b = \pm c$, т. е. имеется 8 направлений (по одному в каждом из октантов), которые удовлетворяют этому условию. Однако существует лишь 4 различных изометрических проекции (если не рассматривать удаление скрытых линий), так как векторы *(a, a, a)* и *(-a,-a,-a)* определяют нормали к одной и той же проекционной плоскости.

Изометрическая проекция (рис. 11.10.) обладает следующим свойством: все 3 главные координатные оси одинаково укорачиваются. Поэтому можно проводить измерения вдоль направления осей с одним и тем же масштабом. Кроме того, главные координатные оси проецируются так, что их проекции составляют равные углы друг с другом (120°).

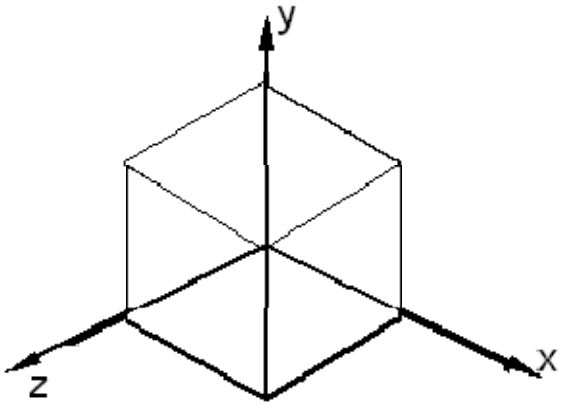


Рис. 11.10. Изометрическая проекция единичного куба

Косоугольные (наклонные) проекции сочетают в себе свойства ортографических проекций (видов спереди, сверху и сбоку) со свойствами аксонометрии. В этом случае проекционная плоскость перпендикулярна главной координатной оси, поэтому сторона объекта, параллельная этой плоскости, проецируется так, что можно измерить углы и расстояния. Проецирование других сторон объекта также допускает проведение линейных измерений (но не угловых) вдоль главных осей. Отметим, что нормаль к проекционной плоскости и направление проецирования не совпадают.

Двумя важными видами косоугольных проекций являются проекции:

- ***Кавалье*** (cavalier) – горизонтальная косоугольная изометрия (военная перспектива);
- ***Кабине*** (cabinet) – фронтальная косоугольная диметрия.

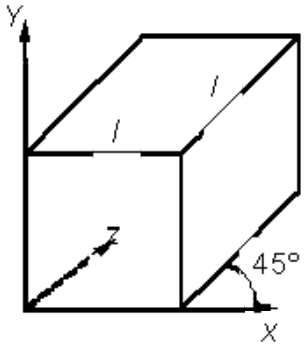


Рис. 11.11. Проекция Кавалье

В проекции ***Кавалье*** (рис. 11.11.) направление проецирования составляет с плоскостью угол 45°. В результате проекция отрезка, перпендикулярного проекционной плоскости, имеет ту же длину, что и сам отрезок, т. е. укорачивание отсутствует.

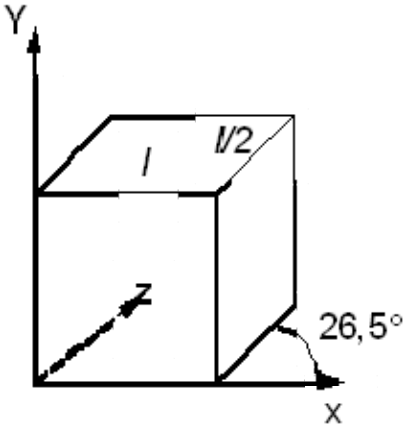


Рис. 11.12. Проекция Кабине

Проекция ***Кабине*** (рис. 11.12.) имеет направление проецирования, которое составляет с проекционной плоскостью угол $\alpha = \operatorname{arctg}(½)$ (≈26,5°). При этом отрезки, перпендикулярные проекционной

плоскости, после проецирования составляют ½ их действительной длины. Проекции Кабине являются более реалистическими, чем проекции Кавалье, так как укорачивание с коэффициентом ½ больше согласуется с нашим визуальным опытом.

Центральная проекция любой совокупности параллельных прямых, которые не параллельны проекционной плоскости, будет сходиться в точке схода. Точек схода бесконечно много. Если совокупность прямых параллельна одной из главных координатных осей, то их точка схода называется ***главной точкой схода***. Имеются только три такие точки, соответствующие пересечениям главных координатных осей с проекционной плоскостью. Центральные проекции классифицируются в зависимости от числа главных точек схода, которыми они обладают, а следовательно и от числа координатных осей, которые пересекают проекционную плоскость.

1. ***Одноточечная*** проекция (рис. 11.13).

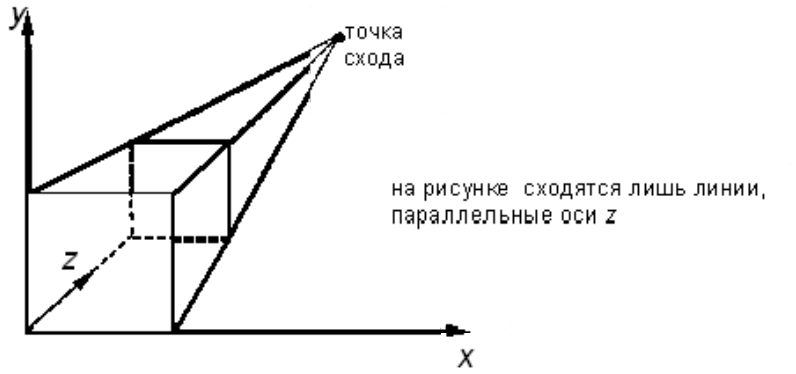


Рис. 11.13. Одноточечная перспектива

2. ***Двухточечная*** проекция широко применяется в архитектурном, инженерном и промышленном проектировании.

3. ***Трехточечные*** центральные проекции почти совсем не используются, во-первых, потому, что их трудно конструировать, а во-

вторых, из-за того, что они добавляют мало нового с точки зрения реалистичности по сравнению с двухточечной проекцией.

11.4. Математическое описание плоских геометрических проекций

Каждую из проекций можно описать матрицей 4×4. Этот способ оказывается удобным, поскольку появляется возможность объединить матрицу проецирования с матрицей преобразования.

Центральная (перспективная) проекция получается путем перспективного преобразования и проецирования на некоторую двухмерную плоскость «наблюдения». Перспективная проекция на плоскость $Z = 0$ обеспечивается преобразованием

$$[X\ Y\ Z\ H] = [x\ y\ z\ 1]^* \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & r \\ 0 & 0 & 0 & 1 \end{bmatrix} = [x\ y\ 0\ (rz+1)].$$

или

$$x^* = \frac{X}{H} = \frac{x}{rz+1};$$

$$y^* = \frac{Y}{H} = \frac{y}{rz+1};$$

$$z^* = \frac{Z}{H} = \frac{0}{rz+1},$$

где

$$r = \frac{1}{k}.$$

Центр проекции находится в точке с координатами *(*0,0,*-k)* (рис.11.14.), плоскость проецирования *Z = 0*.

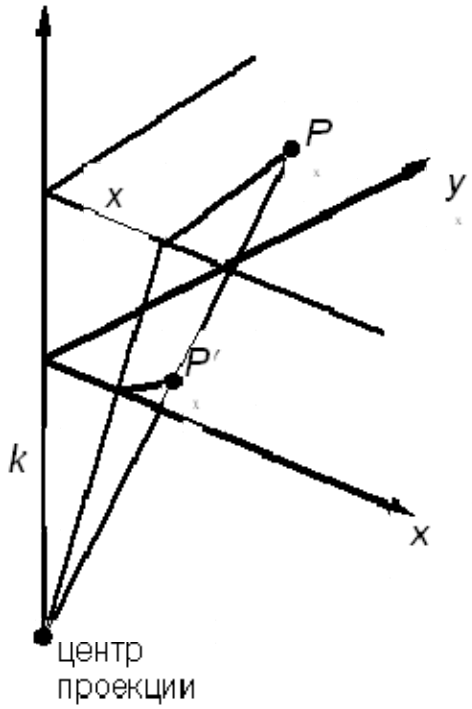


Рис. 11.14. Вычисление одноточечной перспективы

Соотношения между *x, y* и *x*, y** остаются теми же самыми. Рассматривая подобные треугольники, получим, что

$$\frac{x'}{k} = \frac{x}{z+k}, \text{ или } x^* = \frac{x}{\frac{z}{k}+1};$$

аналогично

$$y^* = \frac{y}{\frac{z}{k}+1}.$$

Координаты *x*, y** являются преобразованными координатами. В перспективном проектировании преобразованное пространство не является евклидовым, так как ортогональность осей не сохраняется. При $k = \infty$ получим аксонометрическое преобразование.

Аффинное преобразование есть комбинация линейных преобразований, сопровождаемых переносом.

Последний столбец в обобщенной матрице 4×4 должен быть равен $\begin{bmatrix}0\\0\\0\\1\end{bmatrix}$, в этом случае $H = 1$.

Перспективному преобразованию может предшествовать произвольная последовательность аффинных преобразований. Таким образом, чтобы получить перспективные изображения из произвольной точки наблюдения вначале используют аффинные преобразования, позволяющие сформировать систему координат с осью *Z* вдоль желаемой линии визирования. Затем применяется перспективное преобразование.

Аналогично перспективное преобразование, когда картинная плоскость перпендикулярна оси *Z* и совпадает с плоскостью $Z = 1/r$. Центр проекции находится в центре координат:

$$[X\ Y\ Z\ H] = [x\ y\ z\ 1] * \begin{bmatrix}1 & 0 & 0 & 0\\0 & 1 & 0 & 0\\0 & 0 & 1 & r\\0 & 0 & 0 & 1\end{bmatrix} = [x\ y\ z\ (rz+1)]$$ — одноточечная перспектива (точка схода *Z*);

$$\begin{bmatrix} 1 & 0 & 0 & p \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$ — точка схода *X*.

Двухточечная (угловая) перспектива. Для получения двухточечной перспективы в общей матрице преобразования устанавливают коэффициенты *p* и *q*:

$$(x', y', z', 1) = (x, y, z, 1) \cdot \begin{bmatrix} 1 & 0 & 0 & p \\ 0 & 1 & 0 & q \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} = [x, y, 0, (px+qu+1)];$$

$$(x', y', z', 1) = \left(\frac{x}{px+qy+1}, \frac{y}{px+qy+1}, 0, 1 \right).$$

Такое преобразование приводит к двум точкам схода. Одна расположена на оси *X* в точке $\left(\frac{1}{p}, 0, 0, 1\right)$, другая на оси *Y* в точке $\left(0, \frac{1}{q}, 0, 1\right)$.

Рассмотрим это преобразование на получение проекции единичного куба (рис. 11.15.).

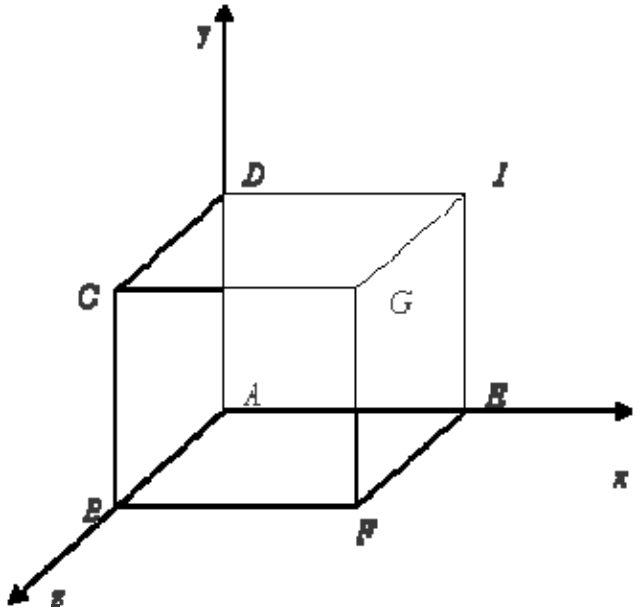


Рис. 11.15. Единичный куб для получения двухточечной проекции

$$\begin{matrix} A \\ B \\ C \\ D \\ E \\ F \\ G \\ I \end{matrix}\begin{bmatrix} 0 & 0 & 0 & 1 \\ 0 & 0 & 1 & 1 \\ 0 & 1 & 1 & 1 \\ 0 & 1 & 0 & 1 \\ 1 & 0 & 0 & 1 \\ 1 & 0 & 1 & 1 \\ 1 & 1 & 1 & 1 \\ 1 & 1 & 0 & 1 \end{bmatrix} \cdot \begin{bmatrix} 1 & 0 & 0 & 0{,}1 \\ 0 & 1 & 0 & 0{,}1 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} = \begin{bmatrix} 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 1 \\ 0 & 1 & 0 & 1{,}1 \\ 0 & 1 & 0 & 1{,}1 \\ 1 & 0 & 0 & 1{,}1 \\ 1 & 0 & 0 & 1{,}1 \\ 1 & 1 & 0 & 1{,}2 \\ 1 & 1 & 0 & 1{,}2 \end{bmatrix} \Rightarrow \begin{bmatrix} 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 1 \\ 0 & 0{,}91 & 0 & 1 \\ 0 & 0{,}91 & 0 & 1 \\ 0{,}91 & 0 & 0 & 1 \\ 0{,}91 & 0 & 0 & 1 \\ 0{,}83 & 0{,}83 & 0 & 1 \\ 0{,}83 & 0{,}83 & 0 & 1 \end{bmatrix} \begin{matrix} A' \\ B' \\ C' \\ D' \\ E' \\ F' \\ G' \\ I' \end{matrix}.$$

В результате получаем проекцию вида, представленного на рис. 11.16.

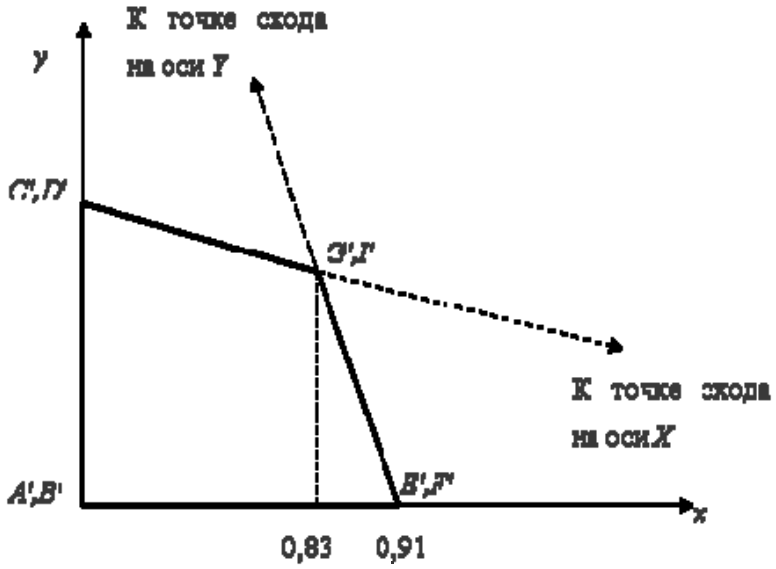


Рис. 11.16. Двухточечная проекция единичного куба

$$\begin{bmatrix} 1 & 0 & 0 & p \\ 0 & 1 & 0 & q \\ 0 & 0 & 1 & r \\ 0 & 0 & 0 & 1 \end{bmatrix}$$ $=[x\ y\ z\ (px+qy+rz+1)]$ — трехточечная (косая) перспектива.

Для того чтобы создать **диметрическую проекцию**, необходимо выполнить следующее условие:

$\sin^2\varphi=\sin^2\theta/(1-\sin^2\theta)$.

Одним способом выбора sinθ является сокращение оси *Z* в фиксированное число раз. При этом единичный вектор на оси *Z*, равный [0 0 1 1], преобразовывается к виду

$[X\ Y\ Z\ H] = [\sin\varphi\ \ -\cos\varphi\cdot\sin\theta\ \ \cos\varphi\cdot\cos\theta\ \ 1]$

или $x^* = \sin\varphi$;

$y^* = -\cos\varphi\ \sin\theta$.

Таким образом, для диметрической проекции получаем

$\varphi = 20{,}705°$:

$\theta = 22{,}208°$.

Для образования **изометрической проекции** нужно в одинаковое число раз сократить все три оси. Для этого необходимо, чтобы выполнялось условие

$\sin^2\varphi=\sin^2\theta/(1-\sin^2\theta)$ и $\sin^2\varphi=(1-2\sin^2\theta)/(1-\sin^2\theta)$.

Таким образом,

$\varphi = 35{,}26439°$;

$\theta = 45°$.

Рассмотрим теперь **косоугольную проекцию** (рис. 11.17.), матрица может быть записана исходя из значений α и *l*.

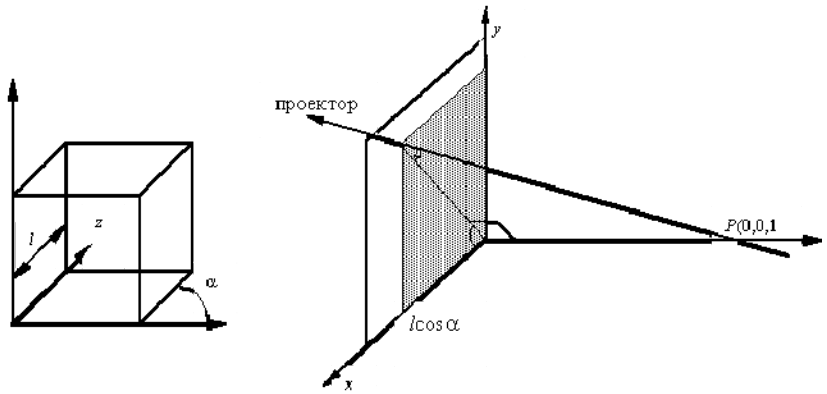


Рис. 11.17. Вычисление косоугольных проекций

Проекцией точки $P(0,0,1)$ является точка $P'(l\cos\alpha,\ l\sin\alpha,\ 0)$, принадлежащая плоскости xy. Направление проецирования совпадает с отрезком PP', проходящим через две эти точки. Это направление есть $P'-P = (l\cos\alpha, l\sin\alpha, -1)$. Направление проецирования составляет угол β с плоскостью xy.

Теперь рассмотрим проекцию точки x, y, z и определим ее косоугольную проекцию $(x_p\ y_p)$ на плоскости xy:

$$x_p = x + z(l\cos\alpha);$$

$$y_p = y + z(l\sin\alpha).$$

Таким образом, матрица 4×4, которая выполняет эти действия и, следовательно, описывает косоугольную проекцию, имеет вид

$$M_{кос} = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ l\cos\alpha & l\sin\alpha & 0 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}.$$

Применение матрицы $M_{кос}$ приводит к сдвигу и последующему проецированию объекта: плоскости с постоянной координатой $z = z_1$ переносятся в направлении x на $z_1\ l\cos\alpha$ и в направлении y на $z_1\ l\sin\alpha$ и затем проецируется на плоскость $z = 0$. Сдвиг сохраняет параллельность прямых, а также углы и расстояния в плоскостях, параллельных оси z.

Для проекции *Кавалье* $l = 1$, поэтому угол b = 45°. Для проекции Кабине l=½, a b = arctg(2) = 63,4°. В случае ортографической проекции $l = 0$ и b = 90°, поэтому матрица ортографического проецирования является частным случаем косоугольной проекции.

11.5. Изображение трехмерных объектов

Процесс вывода трехмерной графической информации более сложный, чем соответствующий двумерный процесс. В двумерном случае просто задается окно в двумерном мировом координатном пространстве и поля вывода на двумерной видовой поверхности. В общем случае объекты, описанные в мировых координатах, отсекаются по границе видимого объема, а после этого преобразуются в поле вывода для дисплея. Сложность, характерная для трехмерного случая, возникает потому, что видовая поверхность не имеет третьего измерения.

Несоответствие между пространственными объектами и плоскими изображениями устраняется путем введения проекций, которые отображают трехмерные объекты на двумерной проекционной ***картинной плоскости*** (КП).

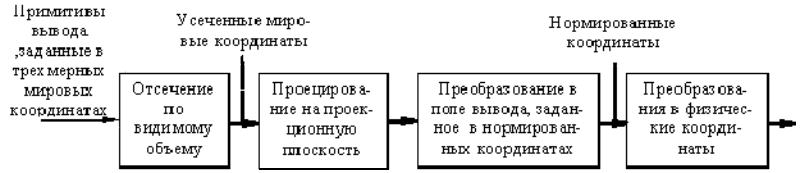


Рис. 11.18. Процесс вывода трехмерной графической информации

В процессе вывода трехмерной графической информации (рис. 11.18.) мы задаем ***видимый объем*** (ВО) в мировом пространстве, проекцию на КП и поле вывода на видовой поверхности. В общем случае объекты,

определенные в трехмерном мировом пространстве, отсекаются по границам трехмерного видимого объема и после этого проецируются. То, что попадает в пределы окна, которое само является проекцией видимого объема на картинную плоскость, затем преобразуется (отображается) в поле вывода и отображается на графическом устройстве.

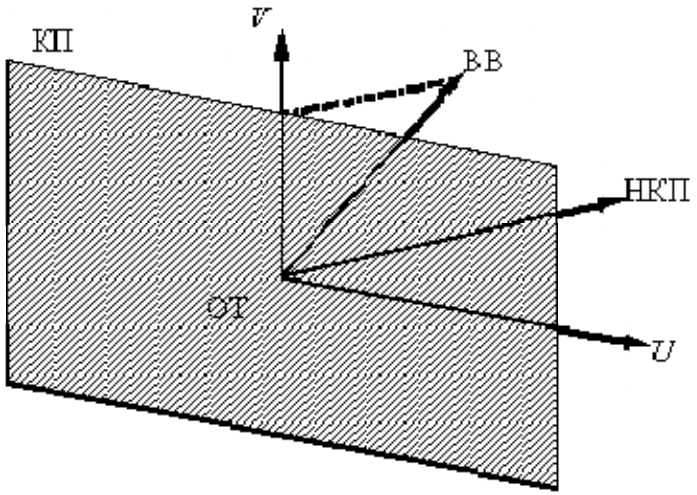


Рис. 11.19. Картинная плоскость и определяющие ее параметры

Картинная плоскость определяется (рис. 11.19.) некоторой точкой на плоскости, которую будем называть ***опорной точкой*** (ОТ) и ***нормалью к картинной плоскости*** (НКП). КП может произвольным образом располагаться относительно проецируемых объектов, заданных в мировых координатах. Она может пересекать их, проходить впереди или позади объектов.

Для того чтобы задать окно, нам необходима система координат на КП, которую назовем системой координат *UV*. Началом ее служит ОТ. Направление оси *V* на КП определяет ***вектор вертикали*** (ВВ): проекция ВВ на КП совпадает с осью *V*.

ОТ и два направления вектора НКП и ВВ определяются в правосторонней мировой системе координат. Имея на КП систему *UV*,

можем задать минимальное и максимальное значения *U* и *V*, определяющие окно (рис. 11.20.).

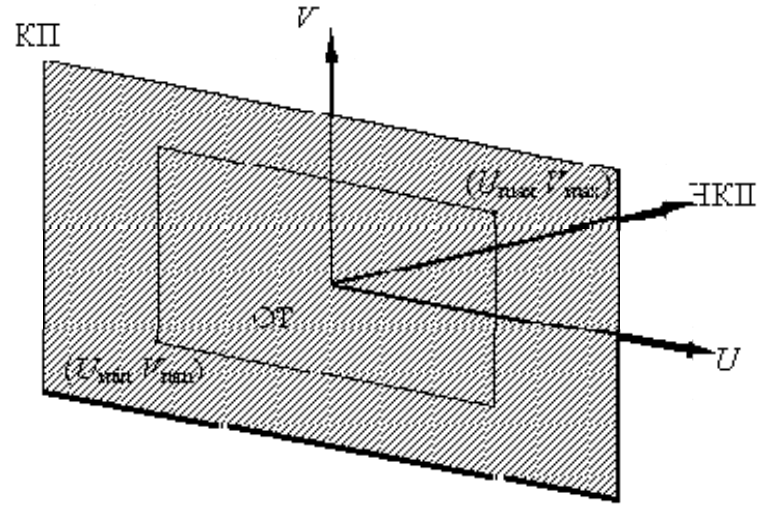


Рис. 11.20. Окно вывода на картинной плоскости

Отметим, что окно не обязательно должно быть симметрично относительно ОТ.

Видимый объем частично определяется окном и ограничивает ту часть мирового пространства, которая будет спроецирована.

В случае центральной проекции ВО определяется также центром проекции (рис. 11.21.). Этот параметр задается в мировых координатах относительно ОТ. ВО представляет собой неограниченную в одну сторону пирамиду, вершина которой находится в центре проекции, а боковые стороны проходят через окно.

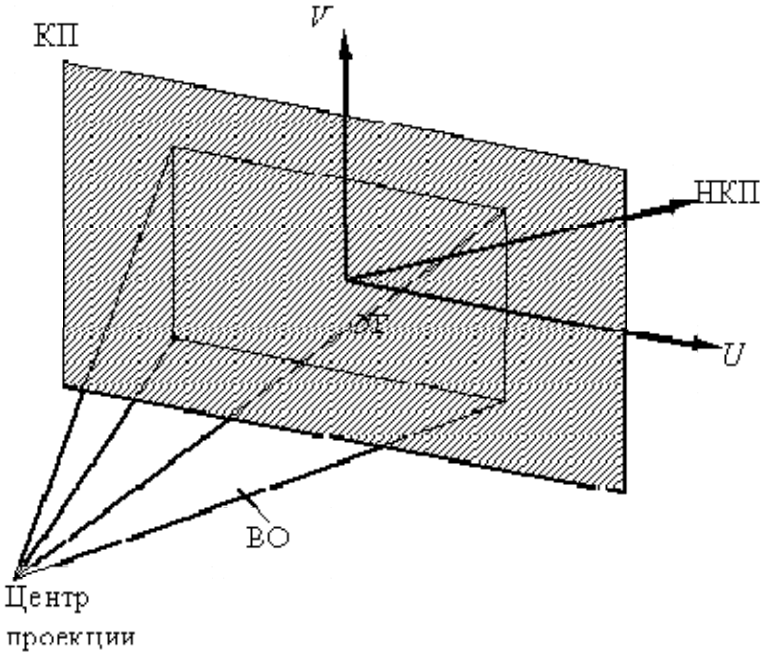


Рис. 11.21. Видимый объем для центральной проекции
Точки, лежащие позади центра проекции, не включаются в ВО и, следовательно, не будут проецироваться.

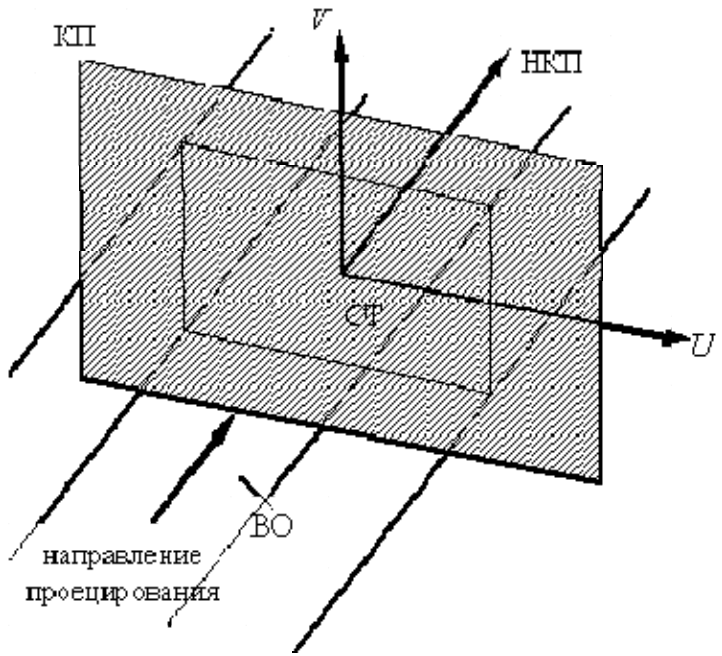


Рис. 11.22. Видимый объем параллельной проекции
В случае параллельных проекций ВО определяется также направлением проецирования (рис. 11.22.). Он представляет собой неограниченный параллелепипед, стороны которого параллельны направлению проецирования.

В общем случае направление проецирования может не совпадать с НКП.

В случае ортографических параллельных проекций (но не косоугольных) боковые стороны ВО перпендикулярны КП.

В некоторых случаях может потребоваться сделать ВО конечным (рис. 11.23.-11.25.). Для этого задаются ПСП (передняя секущая плоскость) и ЗСП (задняя секущая плоскость).

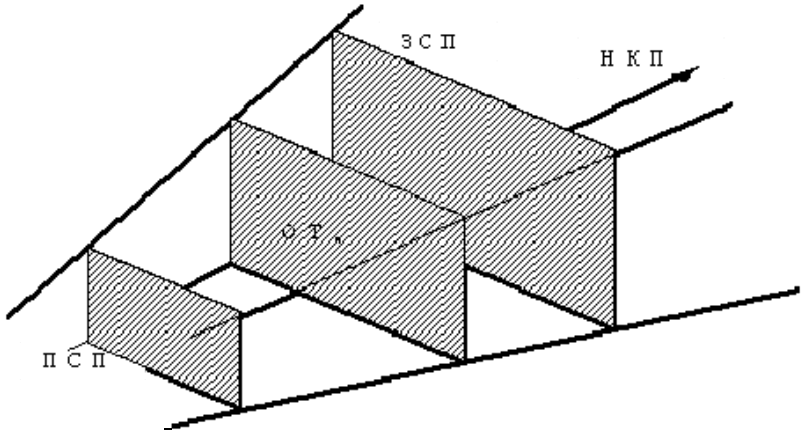


Рис. 11.23. Усеченный ВО для центральной проекции

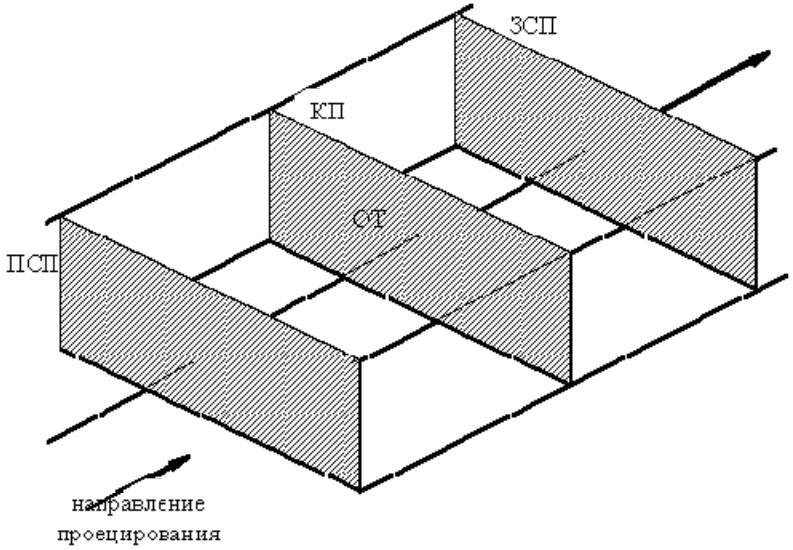


Рис. 11.24. Усеченный ВО для ортографической параллельной проекции

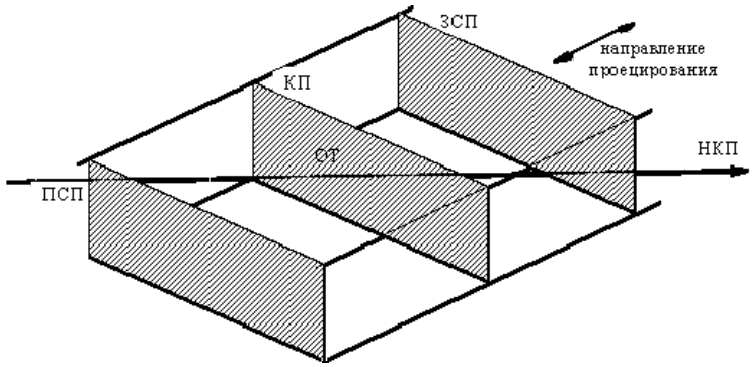


Рис. 11.25. Усеченный ВО для косоугольной параллельной проекции.
Нормаль НКП направлена относительно направления проецирования и также является нормалью к ПСП и ЗСП.

12. Представление пространственных форм

В системах ИИ часто возникает потребность в представлении трехмерных форм: при проектировании самолетов, при восстановлении трехмерных тел по изображениям их поперечных сечений, построенных с помощью машинной томографии, при автоматической сборке и во многих других. Нам уже известно, как изображаются пространственные объекты, когда их удается представить в виде последовательности отрезков прямых, заданных в мировых координатах. **Совокупность отрезков не является адекватным описанием объекта, поскольку отрезки сами по себе не определяют поверхностей.** В то же время информация о поверхностях необходима для проведения вычислений, связанных со стиранием скрытых частей изображения, для определения объемов и т. д. Таким образом, мы приходим к выводу, что **для описания трехмерных форм необходимы поверхности – примитивы более высокого уровня, чем отрезки.**

Мы остановим внимание на двух широко распространенных трехмерных представлениях поверхностей в пространстве: **полигональных сетках и параметрических бикубических кусках**. ***Полигональной сеткой*** **является совокупность связанных между**

собой плоских многоугольников. Наружную форму большинства зданий можно легко и естественно описать с помощью полигональной сетки (так же, как мебель и комнаты). Полигональные сетки применяются также для представления объектов, ограниченных криволинейными поверхностями. Однако недостатком этого метода является его приблизительность. Видимые ошибки в таком представлении можно сделать сколь угодно малыми, используя все большее число многоугольников для улучшения кусочно-линейной аппроксимации объекта, но это приведет к дополнительным затратам памяти и вычислительного времени для алгоритмов, работающих с таким представлением.

Параметрические бикубические куски **описывают координаты точек на искривленной поверхности с помощью трех уравнений (по одному для *x*, *y* и *z*). Каждое из уравнений имеет две переменные (два параметра), причем показатели степени при них не выше третьей (отсюда название бикубический). Границами кусков являются параметрические кубические кривые.** Для представления поверхности с заданной точностью требуется значительно меньшее число бикубических кусков, чем при аппроксимации полигональной сеткой. Однако алгоритмы для работы с бикубическими объектами существенно сложнее алгоритмов, имеющих дело с многоугольниками.

При использовании обоих методов **трехмерное тело представляется в виде замкнутой поверхности.**

12.1. Полигональные сетки

Полигональная сетка представляет собой совокупность ребер, вершин и многоугольников. Вершины соединяются ребрами, а многоугольники рассматриваются как последовательности ребер или вершин. Сетку можно представить несколькими различными способами, каждый из них имеет свои достоинства и недостатки. Для оценки оптимальности представления используют следующие критерии:

- Объем требуемой памяти;
- Простота идентификации ребер, инцидентных вершине;

- Простота идентификации многоугольников, которым принадлежит данное ребро;
- Простота процедуры поиска вершин, образующих ребро;
- Легкость определения всех ребер, образующих многоугольник;
- Простота получения изображения полигональной сетки;
- Простота обнаружения ошибок в представлении (например, отсутствие ребра или вершины или многоугольника).

Рассмотрим подробнее три способа описания полигональных сеток.

12.1.1. Явное задание многоугольников

Каждый многоугольник можно представить в виде списка координат его вершин:

$$P = ((x_1, y_1, z_1), (x_2, y_2, z_2), \ldots, (x_n, y_n, z_n))$$

Вершины запоминаются в том порядке, в котором они встречаются при обходе вокруг многоугольника. При этом все последовательные вершины многоугольника (а также первая и последняя) соединяются ребрами. Для каждого отдельного многоугольника данный способ записи является эффективным, однако для полигональной сетки дает потери памяти вследствие дублирования информации о координатах общих вершин.

Полигональная сетка изображается путем вычерчивания ребер каждого многоугольника, однако это приводит к тому, что общие ребра рисуются дважды – по одному разу для каждого из многоугольников.

12.1.2. Задание многоугольников с помощью указателей в список вершин

При использовании этого представления каждый узел полигональной сетки запоминается лишь один раз в списке вершин $V=((x_1, y_1, z_1), \ldots,$

(x_n, y_n, z_n)). Многоугольник определяется списком указателей (или индексов) в списке вершин. Многоугольник, составленный из вершин 3, 5, 7 и 10 этого списка, представляется как P = (3, 5, 7, 10).

Такое представление имеет ряд преимуществ по сравнению с явным заданием многоугольников. Поскольку каждая вершина многоугольника запоминается только один раз, удается сэкономить значительный объем памяти. Кроме того, координаты вершины можно легко изменять. Однако все еще не просто отыскивать многоугольники с общими ребрами. Ребра при изображении всей полигональной фигуры по-прежнему рисуются дважды. Эти две проблемы можно решить, если описывать ребра в явном виде.

12.1.3. Явное задание ребер

В этом представлении имеется список вершин V, однако будем рассматривать теперь многоугольник как совокупность указателей на элементы списка ребер, в котором ребра встречаются лишь один раз. Каждое ребро в списке ребер указывает на две вершины в списке вершин, определяющие это ребро, а также на один или два многоугольника, которым это ребро принадлежит. Таким образом, мы описываем многоугольник как P =(E_1, ..., E_2), а ребро — как E = (V_1, V_2, P_1, P_2). Если ребро принадлежит только одному многоугольнику, то либо P_1 либо P_2 – пусто.

При явном задании ребер полигональная сетка изображается путем вычерчивания не всех многоугольников, а всех ребер. В результате удается избежать многократного рисования общих ребер. Отдельные многоугольники при этом также изображаются довольно просто.

В некоторых приложениях ребра полигональных сеток являются общими для более чем двух многоугольников. Рассмотрим, например, случай в картографии, когда такие подразделения, как округа, штаты и т. д., описываются многоугольниками. Ребро (или последовательность ребер), представляющее часть границы между двумя штатами, является также границей округа в каждом штате, а возможно, и города. Таким образом, ребро может принадлежать одновременно шести многоугольникам. Если принять во внимание деление городов на районы, избирательные округа и школьные участки, то это число соответственно возрастет. Для таких приложений описания ребер

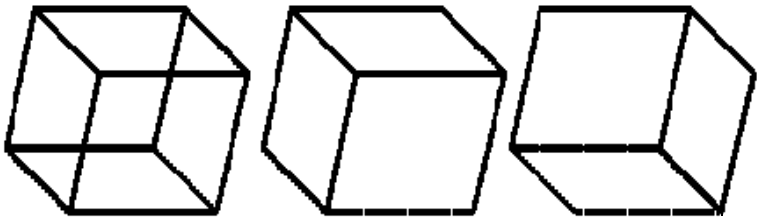
могут быть расширены, чтобы включить произвольное число многоугольников: E = (V_1, V_2, P_1, P_2, …, P_n).

Ни в одном из этих представлений задача определения ребер, инцидентных вершине, не является простой: для ее решения необходимо перебрать все ребра. Конечно, для определения таких отношений можно непосредственно использовать дополнительную информацию.

Удаление невидимых линий и поверхностей

13.1. Введение

Задача удаления невидимых линий и поверхностей является одной из наиболее сложных в машинной графике. Алгоритмы удаления невидимых линий и поверхностей служат для определения линий ребер, поверхностей или объемов, которые видимы или невидимы для наблюдателя, находящегося в заданной точке пространства. Необходимость удаления невидимых линий, ребер, поверхностей или объемов проиллюстрирована рис. 13.1.



а б в

Рис.13.1. Необходимость удаления невидимых линий

На рис. 13.1., а приведен типичный каркасный чертеж куба. Каркасный чертеж представляет трехмерный объект в виде штрихового изображения его ребер. Рис. 13.1. а можно интерпретировать двояко: как вид куба сверху слева или снизу справа. Для этого достаточно прищуриться и перефокусировать глаза. Удаление тех линий или поверхностей, которые невидимы с соответствующей точки зрения, позволяют избавиться от неоднозначности. Результаты показаны на рис. 13.1., б и 13.1., в.

Сложность задачи удаления невидимых линий и поверхностей привела к появлению большого числа различных способов ее решения. Многие из них ориентированы на специализированные приложения. **Наилучшего решения обшей задачи удаления невидимых линий и поверхностей не существует.** Для моделирования процессов в

реальном времени, например для авиатренажеров, требуются быстрые алгоритмы, которые могут порождать результаты с частотой видеогенерации 30 кадр/с. Для машинной мультипликации, например, требуются алгоритмы, которые могут генерировать сложные реалистические изображения, в которых представлены тени, прозрачность и фактура, учитывающие эффекты отражения и преломления цвета в мельчайших оттенках. Подобные алгоритмы работают медленно, и зачастую на вычисления требуется несколько минут или даже часов. Строго говоря, учет эффектов прозрачности, фактуры, отражения и т. п. не входит в задачу удаления невидимых линий или поверхностей. Естественнее считать их частью процесса визуализации изображения. Однако многие из этих эффектов встроены в алгоритмы удаления невидимых поверхностей. **Существует тесная взаимосвязь между скоростью работы алгоритма и детальностью его результата.** Ни один из алгоритмов не может достигнуть хороших оценок для этих двух показателей одновременно. По мере создания все более быстрых алгоритмов можно строить все более детальные изображения. Реальные задачи, однако, всегда будут требовать учета еще большего количества деталей.

Все алгоритмы удаления невидимых линий (поверхностей) включают в себя сортировку. Порядок, в котором производится сортировка координат объектов, вообще говоря, не влияет на эффективность этих алгоритмов**. Главная сортировка ведется по геометрическому расстоянию от тела, поверхности, ребра или точки до точки наблюдения. Основная идея, положенная в основу сортировки по расстоянию, заключается в том, что чем дальше расположен объект от точки наблюдения, тем больше вероятность, что он будет полностью или частично заслонен одним из объектов, более близких к точке наблюдения.** После определения расстояний или приоритетов по глубине остается провести сортировку по горизонтали и по вертикали, чтобы выяснить, будет ли рассматриваемый объект действительно заслонен объектом, расположенным ближе к точке наблюдения. **Эффективность любого алгоритма удаления невидимых линий или поверхностей в большой мере зависит от эффективности процесса сортировки. Для повышения эффективности сортировки используется также когерентность сцены, т. е. тенденция неизменяемости характеристик сцены в малом.**

Алгоритмы удаления невидимых линий или поверхностей можно классифицировать по способу выбора системы координат или

пространства, в котором они работают. Выделяют три класса алгоритмов удаления невидимых линий или поверхностей:

- Алгоритмы, работающие в объектном пространстве.
- Алгоритмы, работающие в пространстве изображения (экрана).
- Алгоритмы, формирующие список приоритетов.

Алгоритмы, работающие в объектном пространстве, имеют дело с физической системой координат, в которой описаны эти объекты. **При этом получаются весьма точные результаты, ограниченные лишь точностью вычислений.** Полученные изображения можно свободно увеличивать во много раз. **Алгоритмы, работающие в объектном пространстве, особенно полезны в тех приложениях, где необходима высокая точность.** Алгоритмы же, работающие в пространстве изображения, имеют дело с системой координат того экрана, на котором объекты визуализируются. При этом точность вычислений ограничена разрешающей способностью экрана. Обычно разрешение экрана бывает довольно низким, типичный пример: 512×512 точек. Результаты, полученные в пространстве изображения, а затем увеличенные во много раз, не будут соответствовать исходной сцене. Например, могут не совпасть концы отрезков. Алгоритмы, формирующие список приоритетов, работают попеременно в обеих упомянутых системах координат. Объем вычислений для любого алгоритма, работающего в объектном пространстве и сравнивающего каждый объект сцены со всеми остальными объектами этой сцены, растет теоретически, как квадрат числа объектов (n^2). Аналогично, объем вычислений любого алгоритма, работающего в пространстве изображения и сравнивающего каждый объект сцены с позициями всех пикселов в системе координат экрана, растет теоретически, как nN. Здесь n обозначает количество объектов (тел, плоскостей или ребер) в сцене, а N — число пикселов. Теоретически трудоемкость алгоритмов, работающих в объектном пространстве, меньше трудоемкости алгоритмов, работающих в пространстве изображения, при $n < N$. Однако на практике это не так. Дело в том, что алгоритмы, работающие в пространстве изображения, более эффективны потому, что для них легче воспользоваться преимуществом когерентности при растровой реализации.

13.2. Алгоритм плавающего горизонта

Алгоритм плавающего горизонта можно отнести к классу алгоритмов, работающих в пространстве изображения. Алгоритм плавающего горизонта чаше всего используется для удаления невидимых линий трехмерного представления функций, описывающих поверхность в виде

$F(x, y, z) = 0$.

Подобные функции возникают во многих приложениях в математике, технике, естественных науках и других дисциплинах.

Главная идея данного метода заключается в сведении трехмерной задачи к двумерной путем пересечения исходной поверхности последовательностью параллельных секущих плоскостей, имеющих постоянные значения координат x, y или z.

На рис. 13.2. приведен пример, где указанные параллельные плоскости определяются постоянными значениями z. Функция $F(x,y,z) = 0$ сводится к последовательности кривых, лежащих в каждой из этих параллельных плоскостей, например к последовательности $y=f(x,z)$ или $x=g(y,z)$, где z постоянно на каждой из заданных параллельных плоскостей.

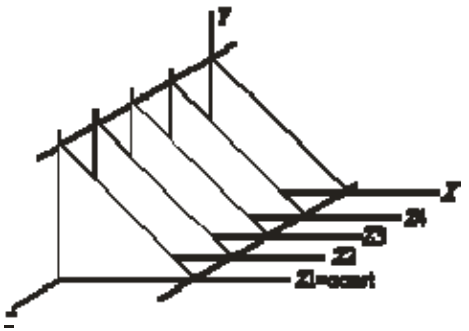


Рис. 13.2. Секущие плоскости с постоянной координатой

Итак, поверхность теперь складывается из последовательности кривых, лежащих в каждой из этих плоскостей, как показано на рис. 13.3.

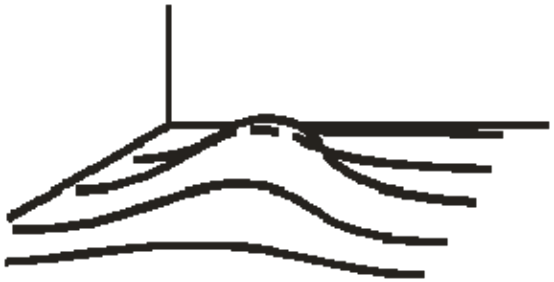


Рис. 13.3. Секущие плоскости с постоянной координатой

Здесь предполагается, что полученные кривые являются однозначными функциями независимых переменных. Если спроецировать полученные кривые на плоскость $z = 0$, как показано на рис.13.4, то сразу становится ясна идея алгоритма удаления невидимых участков исходной поверхности.

Рис. 13.4. Проекция кривых на плоскость z = 0

Алгоритм сначала упорядочивает плоскости z = const по возрастанию расстояния до них от точки наблюдения. Затем для каждой плоскости, начиная с ближайшей к точке наблюдения, строится кривая, лежащая

на ней, т. е. для каждого значения координаты x в пространстве изображения определяется соответствующее значение y. Алгоритм удаления невидимой линии заключается в следующем.

Если на текущей плоскости при некотором заданном значении x соответствующее значение y на кривой больше значения y для всех предыдущих кривых при этом значении x, то текущая кривая видима в этой точке; в противном случае она невидима.

Невидимые участки показаны пунктиром на рис. 13.4. Реализация данного алгоритма достаточно проста. Для хранения максимальных значений y при каждом значении x используется массив, длина которого равна числу различимых точек (разрешению) по оси x в пространстве изображения. Значения, хранящиеся в этом массиве, представляют собой текущие значения "горизонта". Поэтому по мере рисования каждой очередной кривой этот горизонт "всплывает". Фактически этот алгоритм удаления невидимых линий работает каждый раз с одной линией.

Алгоритм работает очень хорошо до тех пор, пока какая-нибудь очередная кривая не окажется ниже самой первой из кривых, как показано на рис. 13.5, а.

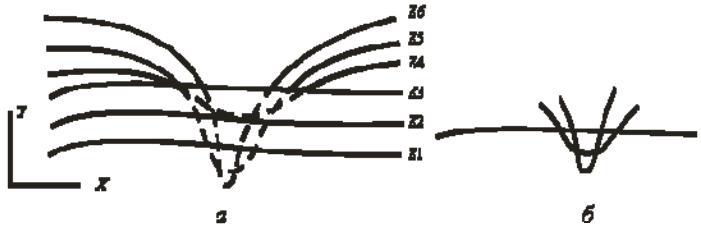


Рис. 13.5. Обработка нижней стороны поверхности

Подобные кривые, естественно, видимы и представляют собой нижнюю сторону исходной поверхности, однако алгоритм будет считать их невидимыми. Нижняя сторона поверхности делается

видимой, если модифицировать этот алгоритм, включив в него нижний горизонт, который опускается вниз по ходу работы алгоритма. Это реализуется при помощи второго массива, длина которого равна числу различимых точек по оси *x* в пространстве изображения. Этот массив содержит наименьшие значения *y* для каждого значения *x*. Алгоритм теперь становится таким: если на текущей плоскости при некотором заданном значении *x* соответствующее значение *y* на кривой больше максимума или меньше минимума по *y* для всех предыдущих кривых при этом значении *x*, то текущая кривая видима. В противном случае она невидима. Полученный результат показан на рис. 13.5, б.

В изложенном алгоритме предполагается, что значение функции, т. е. *y*, известно для каждого значения *x* в пространстве изображения. Однако если для каждого значения *x* нельзя указать (вычислить) соответствующее ему значение y, то невозможно поддерживать массивы верхнего и нижнего плавающих горизонтов. В таком случае используется линейная интерполяция значений *y* между известными значениями для того, чтобы заполнить массивы верхнего и нижнего плавающих горизонтов.

Изложенный алгоритм приводит к некоторым дефектам, когда кривая, лежащая в одной из более удаленных от точки наблюдения плоскостей, появляется слева или справа из-под множества кривых, лежащих в плоскостях, которые ближе к указанной точке наблюдения. Этот эффект продемонстрирован на рис. 13.6, где уже обработанные плоскости *n*-1 и *n* расположены ближе к точке наблюдения. На рисунке показано, что получается при обработке плоскости *n*+1. После обработки кривых *n*-1 и n верхний горизонт для значений $x = 0$ и $x = 1$ равен начальному значению *y*; для значений *x* от 2 до 17 он равен ординатам кривой *n*; а для значений 18, 19, 20 - ординатам кривой *n*-1. Нижний горизонт для значений $x = 0$ и $x = 1$ равен начальному значению *y*; для значений $x = 2, 3, 4$ - ординатам кривой *n*; а для значений *x* от 5 до 20 - ординатам кривой *n*-1. При обработке текущей кривой (*n*+1) алгоритм объявляет ее видимой при $x = 4$. Это показано сплошной линией на рис. 13.6. Аналогичный эффект возникает и справа при $x = 18$. Такой эффект приводит к появлению зазубренных боковых ребер. Проблема с зазубренностью боковых ребер решается включением в массивы верхнего и нижнего горизонтов ординат, соответствующих штриховым линиям на рис. 13.6. Это можно выполнить эффективно, создав ложные боковые ребра.

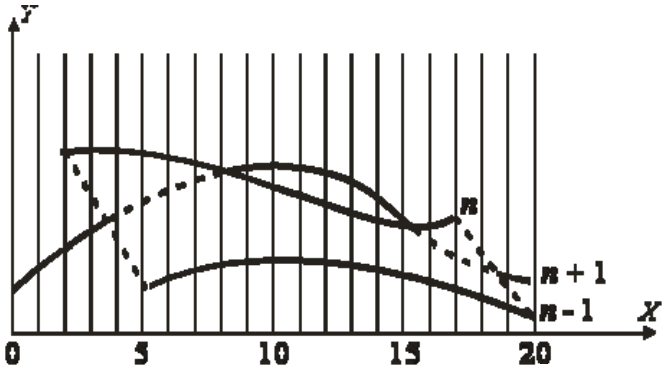


Рис. 13.6. Эффект зазубренного ребра

Если функция содержит очень острые участки (пики), то приведенный алгоритм также может дать некорректные результаты. Этот эффект вызван вычислением значений функции и оценкой ее видимости на участках, меньших, чем разрешающая способность экрана, т. е. тем, что функция задана слишком малым количеством точек. Если встречаются узкие участки, то функцию следует вычислять в большем числе точек. На рис. 13.7 показан типичный результат работы алгоритма плавающего горизонта для функции $y = (1/5)\sin x \cos z - (3/2) \cos (7\alpha/4) e^{(-\alpha)}$, где $\alpha = (x - \pi)^2+(z - \pi)^2$, в интервале $(0, 2\pi)$.

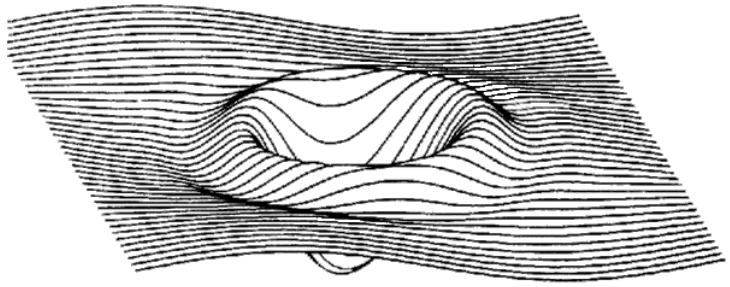


Рис. 13.7. Результат работы алгоритма плавающего горизонта

13.3. Алгоритм Робертса

Алгоритм Робертса представляет собой первое известное решение задачи об удалении невидимых линий. Это математически элегантный метод, работающий в объектном пространстве. Алгоритм прежде всего удаляет из каждого тела те ребра или грани, которые экранируются самим телом. Затем каждое из видимых ребер каждого тела сравнивается с каждым из оставшихся тел для определения того, какая его часть или части, если таковые есть, экранируются этими телами. Поэтому вычислительная трудоемкость алгоритма Робертса растет теоретически, как квадрат числа объектов. Это в сочетании с ростом интереса к растровым дисплеям, работающим в пространстве изображения, привело к снижению интереса к алгоритму Робертса. Однако математические методы, используемые в этом алгоритме, просты, мощны и точны. Кроме того, этот алгоритм можно использовать для иллюстрации некоторых важных концепций. Наконец, более поздние реализации алгоритма, использующие предварительную приоритетную сортировку вдоль оси z и простые габаритные или минимаксные тесты, демонстрируют почти линейную зависимость от числа объектов.

Работа Алгоритм Робертса проходит в два этапа:

1. Определение нелицевых граней для каждого тела отдельно.

2. Определение и удаление невидимых ребер.

13.3.1. Определение нелицевых граней

Пусть F — некоторая грань многогранника. Плоскость, несущая эту грань, разделяет пространство на два подпространства. Назовем положительным то из них, в которое смотрит внешняя нормаль к грани. Если точка наблюдения – в положительном подпространстве, то грань – **лицевая**, в противном случае – **нелицевая**. Если многогранник выпуклый, то удаление всех нелицевых граней полностью решает задачу визуализации с удалением невидимых граней.

Для определения, лежит ли точка в положительном подпространстве, используют проверку знака скалярного произведения (l, n), где l –

вектор, направленный к наблюдателю, фактически определяет точку наблюдения; n – вектор внешней нормали грани. Если $(l, n) > 0$, т. е. угол между векторами острый, то грань является лицевой. Если $(l, n) < 0$, т. е. угол между векторами тупой, то грань является нелицевой.

В алгоритме Робертса требуется, чтобы все изображаемые тела или объекты были выпуклыми. Невыпуклые тела должны быть разбиты на выпуклые части. В этом алгоритме выпуклое многогранное тело с плоскими гранями должно представиться набором пересекающихся плоскостей. Уравнение произвольной плоскости в трехмерном пространстве имеет вид

$$ax + by + cz + d = 0 \qquad (13.1.)$$

В матричной форме этот результат выглядит так:

$$[x\ y\ z\ 1][P]^T = 0,$$

где $[P]^T = [a\ b\ c\ d]$ представляет собой плоскость. Поэтому любое выпуклое твердое тело можно выразить матрицей тела, состоящей из коэффициентов уравнений плоскостей, т. е.

$$[V] = \begin{bmatrix} a_1 & a_2 & \dots & a_n \\ b_1 & b_2 & \dots & b_n \\ c_1 & c_2 & \dots & c_n \\ d_1 & d_2 & \dots & d_n \end{bmatrix},$$

где каждый столбец содержит коэффициенты одной плоскости.

Напомним, что любая точка пространства представима в однородных координатах вектором $[S] = [x\ y\ z\ 1]$. Более того, если точка $[S]$ лежит на плоскости, то $[S]\cdot[P]^T = 0$. Если же $[S]$ не лежит на плоскости, то знак этого скалярного произведения показывает, по какую сторону от плоскости расположена точка. В алгоритме Робертса предполагается, что точки, лежащие внутри тела, дают отрицательное скалярное

произведение, т. е. нормали направлены наружу. Чтобы проиллюстрировать эти идеи, рассмотрим следующий пример.

Рассмотрим единичный куб с центром в начале координат (рис.13.8).

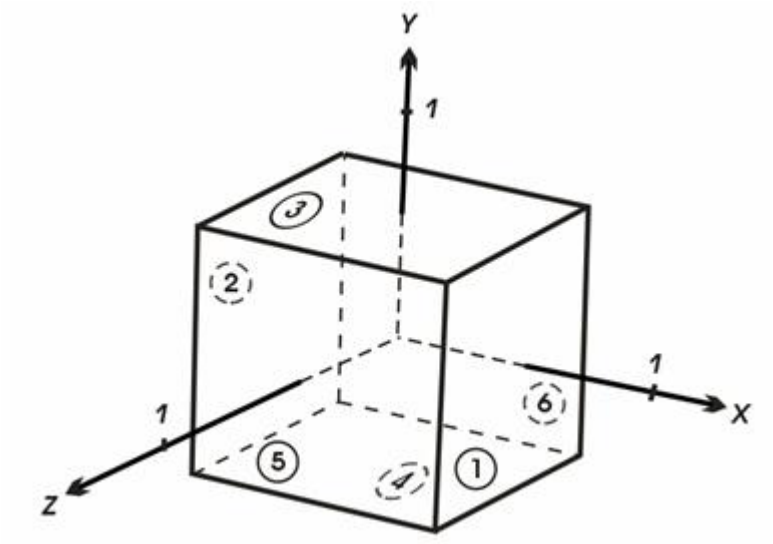


Рис. 13.8. Куб с центром в начале координат

Шесть плоскостей, описывающих данный куб, таковы:

$x_1 = 1/2, x_2 = -1/2, \ y_3 = 1/2, \ y_4 = -1/2, z_5 = 1/2, z_6 = -1/2.$

Более подробно уравнение правой плоскости можно записать как

$x_1 + 0 \cdot y_1 + 0 \cdot z_1 - (1/2) = 0$ или $2x_1 - 1 = 0$

Полная матрица тела такова:

$$[V] = \begin{bmatrix} 1 & 1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 1 & 0 \\ 0 & 0 & 0 & 0 & 1 \\ -1/2 & 1/2 & -1/2 & 1/2 & -1/2 \end{bmatrix} = \begin{bmatrix} 2 & 2 & 0 & 0 & 0 \\ 0 & 0 & 2 & 2 & 0 \\ 0 & 0 & 0 & 0 & 2 \\ -1 & 1 & -1 & 1 & -1 \end{bmatrix}.$$

Экспериментально проверим матрицу тела с помощью точки, о которой точно известно, что она лежит внутри тела. Если знак скалярного произведения для какой-нибудь плоскости больше нуля, то соответствующее уравнение плоскости следует умножить на -1. Для проверки возьмем точку внутри куба с координатами x = 1/4, y = 1/4, z = 1/4. В однородных координатах эта точка представляется в виде вектора

$$[S] = [1/4 \ \ 1/4 \ \ 1/4 \ \ 1] = [1 \ \ 1 \ \ 1 \ \ 4].$$

Скалярное произведение этого вектора на матрицу объема равно

$$[S] \cdot [V] = [1 \ 1 \ 1 \ 4] \cdot \begin{bmatrix} 1 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 1 \\ -1/2 & 1/2 & -1/2 & 1/2 & -1/2 & 1/2 \end{bmatrix} = [-2 \ \ 6 \ \ -2 \ \ 6 \ \ -2 \ \ 6].$$

Здесь результаты для второго, четвертого и шестого уравнения плоскостей (столбцов) положительны и, следовательно, составлены некорректно. Умножая эти уравнения (столбцы) на -1, получаем корректную матрицу тела для куба:

$$[V] = \begin{bmatrix} 2 & -2 & 0 & 0 & 0 & 0 \\ 0 & 0 & 2 & -2 & 0 & 0 \\ 0 & 0 & 0 & 0 & 2 & -2 \\ -1 & -1 & -1 & -1 & -1 & -1 \end{bmatrix},$$

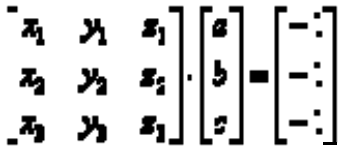
В приведенном примере корректность уравнений плоскостей была проверена экспериментально. Разумеется, это не всегда возможно. Существует несколько полезных методов для более общего случая. Хотя уравнение плоскости содержит четыре неизвестных коэффициента, его можно нормировать так, чтобы d = 1. Следовательно, трех неколлинеарных точек достаточно для определения этих коэффициентов. Подстановка координат трех неколлинеарных точек (x_1, y_1, z_1), (x_2, y_2, z_2), (x_3, y_3, z_3) в нормированное уравнение (13.1.) дает

$$ax_1 + by_1 + cz_1 = -1;$$

$$ax_2 + by_2 + cz_3 = -1;$$

$$ax_3 + by_2 + cz_3 = -1.$$

В матричной форме это выглядит так:



или

$$[X][C] = [D] \qquad (13.2.)$$

Решение этого уравнения дает значения коэффициентов уравнения плоскости: $[C] = [X]^{-1}[D]$.

Другой способ используется, если известен вектор нормали к плоскости, т. е.

$$n = a\boldsymbol{i} + b\boldsymbol{j} + c\boldsymbol{k},$$

где ***i, j, k*** – единичные векторы осей x, y, z соответственно. Тогда уравнение плоскости примет вид

$$ax + by + cz + d = 0 \qquad (13.3.)$$

Величина d вычисляется с помощью произвольной точки на плоскости. В частности, если компоненты этой точки на плоскости (x_1, y_1, z_1), то

$$d = -(ax_1 + by_1 + cz_1) \qquad (13.4.)$$

Перед началом работы алгоритма удаления невидимых линий или поверхностей для получения желаемого вида сцены часто применяется трехмерное видовое преобразование. Матрицы тел для объектов преобразованной сцены можно получить или преобразованием исходных матриц тел, или вычислением новых матриц тел, используя преобразованные вершины или точки.

Если $[B]$ – матрица однородных координат, представляющая исходные вершины тела, а $[T]$ – матрица размером 4×4 видового преобразования, то преобразованные вершины таковы:

$$[BT] = [B][T], \qquad (13.5.)$$

где $[BT]$ – преобразованная матрица вершин. Использование уравнения (13.2.) позволяет получить уравнения исходных плоскостей, ограничивающих тело:

$$[B][V] = [D], \qquad (13.6.)$$

где $[V]$ – матрица тела, а $[D]$ в правой части – нулевая матрица. Аналогично уравнения преобразованных плоскостей задаются следующим образом:

$$[BT][VT] = [D], \qquad (13.7.)$$

где $[VT]$ – преобразованная матрица тела. Приравнивая левые части уравнения (13.6.) и (13.7.), получаем

$[BT][VT] = [B][V]$.

Подставляя уравнение (13.5.), сокращая на $[B]$ и умножая слева на $[T]^{-1}$, имеем

$$[VT] = [T]^{-1}[V].$$

Итак, преобразованная матрица тела получается умножением исходной матрицы тела слева на обратную матрицу видового преобразования.

Тот факт, что плоскости имеют бесконечную протяженность и что скалярное произведение точки (вектора точки) на матрицу тела положительно, если точка лежит вне этого тела, позволяет предложить метод, в котором матрица тела используется для определения граней, которые экранируются самим этим телом.

Положительное скалярное произведение дает только такая плоскость (столбец) в матрице тела, относительно которой точка лежит снаружи, т. е. в положительном подпространстве. Проиллюстрируем это на примере уже рассмотренного единичного куба (рис. 13.9.):

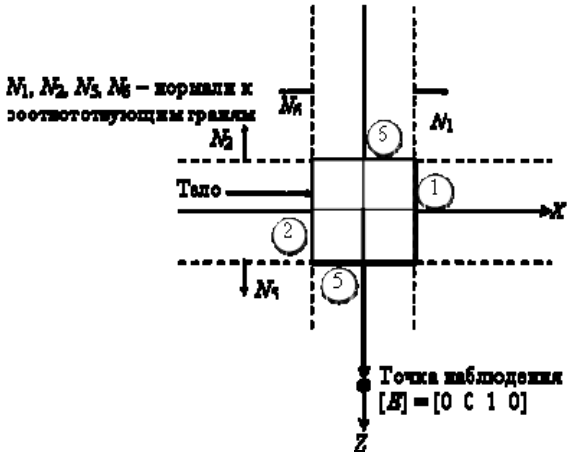


Рис. 13.9. Точка наблюдения вне тела

Условие $[E]\cdot[V] < 0$ определяет, что плоскости — нелицевые, а их грани — задние. Заметим, что для аксонометрических проекций (точка наблюдения в бесконечности) это эквивалентно поиску положительных значений в третьей строке матрицы тела.

Найдем произведение $[E]\cdot[V]$ для нашего примера (рис. 13.9.):

$$[E]\cdot[V] = [0\ 0\ 1\ 0]\cdot\begin{bmatrix} 2 & -2 & 0 & 0 & 0 & 0 \\ 0 & 0 & 2 & -2 & 0 & 0 \\ 0 & 0 & 0 & 0 & 2 & -2 \\ -1 & -1 & -1 & -1 & -1 & -1 \end{bmatrix} = [0\ 0\ 0\ 0\ 2\ -2].$$

Отрицательное число в шестом столбце показывает, что грань с этим номером нелицевая. Положительное число в пятом столбце показывает, что грань лицевая. Нулевые результаты соответствуют плоскостям, параллельным направлению взгляда.

Положительный результат вектора E и вектора нормали можно интерпретировать как острый угол между этими векторами, отрицательный результат – как тупой угол. Если угол между векторами острый, то грань является лицевой; если угол тупой, то грань – нелицевая.

Этот метод является простейшим алгоритмом удаления невидимых поверхностей для тел, представляющих собой одиночные выпуклые многогранники. Он также используется для удаления нелицевых или задних граней из сцены перед применением одного из алгоритмов удаления невидимых линий, которые обсуждаются ниже. Этот способ часто называют *отбрасыванием задних плоскостей*. Для выпуклых многогранников число граней при этом сокращается примерно наполовину. Метод эквивалентен вычислению нормали к поверхности для каждого отдельного многоугольника.

Данный метод определения нелицевых граней в результате формирует аксонометрическую проекцию на некую плоскость, расположенную бесконечно далеко от любой точки трехмерного пространства. Видовые преобразования, включая перспективное, производятся до определения нелицевых плоскостей. Когда видовое преобразование

включает в себя перспективу, то нужно использовать полное перспективное преобразование одного трехмерного пространства в другое, а не перспективное проецирование на некоторую двумерную плоскость. Полное перспективное преобразование приводит к искажению трехмерного тела, которое затем проецируется на некую плоскость в бесконечности, когда нелицевые плоскости уже определены. Этот результат эквивалентен перспективному проецированию из некоторого центра на конечную плоскость проекции.

В литературе описаны и другие способы удаления невидимых граней.

13.3.2. Удаление невидимых ребер

После первого этапа удаления нелицевых отрезков необходимо выяснить, существуют ли такие отрезки, которые экранируются другими телами в картинке или в сцене. Для этого каждый оставшийся отрезок или ребро нужно сравнить с другими телами сцены или картинки.

Возможны следующие случаи:

- Грань ребра не закрывает. Ребро остается в списке ребер.
- Грань полностью закрывает ребро. Ребро удаляется из списка рассматриваемых ребер.
- Грань частично закрывает ребро. В этом случае ребро разбивается на несколько частей, видимыми из которых являются не более двух. Само ребро удаляется из списка рассматриваемых ребер, но в список проверяемых ребер добавляются те его части, которые данной гранью не закрываются.

Для оптимизации используется приоритетная сортировка (*z*-сортировка), а также, сравнения с прямоугольными объемлющими оболочками тел. Такой подход позволяет удалить целые группы или кластеры отрезков и тел. Например, если все тела в сцене упорядочены в некотором приоритетном списке, использующем значения *z* ближайших вершин для представления расстояния до наблюдателя, то

никакое тело из этого списка, у которого ближайшая вершина находится дальше от наблюдателя, чем самая удаленная из концевых точек ребра, не может закрывать это ребро. Более того, ни одно из оставшихся тел, прямоугольная оболочка которого расположена полностью справа, слева, над или под ребром, не может экранировать это ребро. Использование этих приемов значительно сокращает число тел, с которыми нужно сравнивать каждый отрезок или ребро. Рис. 13.10 иллюстрирует работу алгоритма.

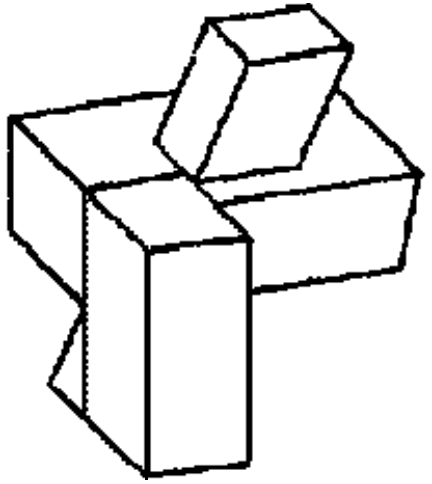


Рис. 13.10. Результат работы алгоритма Робертса

13.4. Алгоритм, использующий z–буфер

Алгоритм, использующий *z-буфер* это один из простейших алгоритмов удаления невидимых поверхностей. Впервые он был предложен Кэтмулом. Работает этот алгоритм в пространстве изображения. Идея *z*-буфера является простым обобщением идеи о буфере кадра. Буфер кадра используется для запоминания атрибутов (интенсивности) каждого пиксела в пространстве изображения, z-буфер - это отдельный буфер глубины, используемый для запоминания координаты *z* или глубины каждого видимого пиксела в пространстве изображения. В процессе работы глубина или значение *z* каждого нового пиксела, который нужно занести в буфер кадра, сравнивается с глубиной того пиксела, который уже занесен в *z*-буфер. Если это сравнение

показывает, что новый пиксел расположен впереди пиксела, находящегося в буфере кадра, то новый пиксел заносится в этот буфер и, кроме того, производится корректировка *z*-буфера новым значением *z*. Если же сравнение дает противоположный результат, то никаких действий не производится. По сути, алгоритм является поиском по *x* и *y* наибольшего значения функции *z (x, y)*.

Главное преимущество алгоритма – его простота. Кроме того, этот алгоритм решает задачу об удалении невидимых поверхностей и делает тривиальной визуализацию пересечений сложных поверхностей. Сцены могут быть любой сложности. Поскольку габариты пространства изображения фиксированы, оценка вычислительной трудоемкости алгоритма не более чем линейна. Поскольку элементы сцены или картинки можно заносить в буфер кадра или в *z*-буфер в произвольном порядке, их не нужно предварительно сортировать по приоритету глубины. Поэтому экономится вычислительное время, затрачиваемое на сортировку по глубине.

Основной недостаток алгоритма - большой объем требуемой памяти. Если сцена подвергается видовому преобразованию и отсекается до фиксированного диапазона значений координат *z*, то можно использовать *z*-буфер с фиксированной точностью. Информацию о глубине нужно обрабатывать с большей точностью, чем координатную информацию на плоскости *(x, y)*; обычно бывает достаточно 20-ти бит. Буфер кадра размером 512×512×24 бит в комбинации с *z*-буфером размером 512×512×20 бит требует почти 1.5 мегабайт памяти. Однако снижение цен на память делает экономически оправданным создание специализированных запоминающих устройств для *z*-буфера и связанной с ним аппаратуры.

Альтернативой созданию специальной памяти для *z*-буфера является использование для этой цели оперативной памяти. Уменьшение требуемой памяти достигается разбиением пространства изображения на 4, 16 или больше квадратов или полос. В предельном варианте можно использовать *z*-буфер размером в одну строку развертки. Для последнего случая имеется алгоритм построчного сканирования. Поскольку каждый элемент сцены обрабатывается много раз, то сегментирование *z*-буфера, вообще говоря, приводит к увеличению времени, необходимого для обработки сцены. Однако сортировка на плоскости, позволяющая не

обрабатывать все многоугольники в каждом из квадратов или полос, может значительно сократить этот рост. Другой недостаток алгоритма *z*-буфера состоит в трудоемкости и высокой стоимости устранения лестничного эффекта, а также реализации эффектов прозрачности и просвечивания. Поскольку алгоритм заносит пикселы в буфер кадра в произвольном порядке, то нелегко получить информацию, необходимую для методов устранения лестничного эффекта, основывающихся на предварительной фильтрации. При реализации эффектов прозрачности и просвечивания пикселы могут заноситься в буфер кадра в некорректном порядке, что ведет к локальным ошибкам.

Формальное описание алгоритма *z*-буфера таково:

1. Заполнить буфер кадра фоновым значением интенсивности или цвета.

2. Заполнить *z*-буфер минимальным значением *z*.

3. Преобразовать каждый многоугольник в растровую форму в произвольном порядке.

4. Для каждого *Пиксел*(*x,y*) в многоугольнике вычислить его глубину *z*(*x,y*).

5. Сравнить глубину *z*(*x,y*) со значением *Zбуфер*(*x,y*), хранящимся в *z*-буфере в этой же позиции.

Если *z*(*x,y*) > *Zбуфер* (*x,y*), то записать атрибут этого многоугольника (интенсивность, цвет и т. п.) в буфер кадра и заменить *Zбуфер*(*x,y*) на *z*(*x,y*). В противном случае никаких действий не производить.

На псевдокоде алгоритм можно представить так:

```
for all objects

    for all covered pixels

        compare z
```

В качестве предварительного шага там, где это целесообразно, применяется удаление нелицевых граней.

Если известно уравнение плоскости, несущей каждый многоугольник, то вычисление глубины каждого пиксела на сканирующей строке можно проделать пошаговым способом. Грань при этом рисуется последовательно (строка за строкой). Для нахождения необходимых значений используется линейная интерполяция (рис. 13.11).

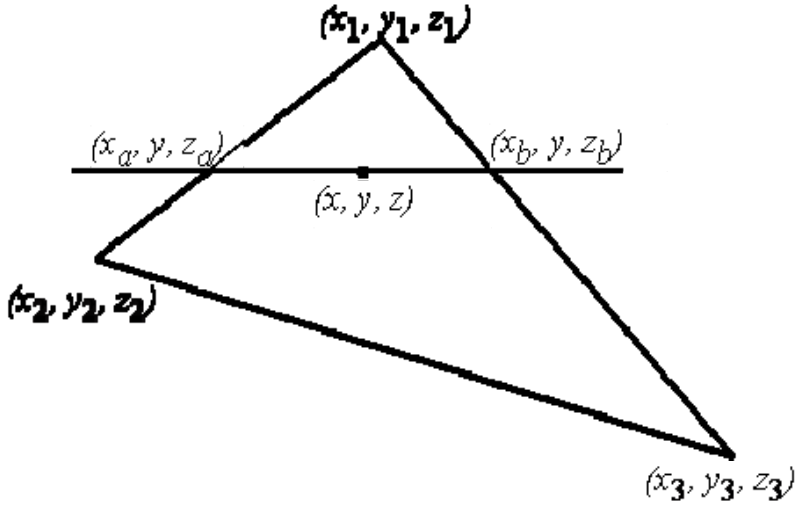


Рис. 13.11. Сканирующая строка по грани
Для рисунка *y* меняется от y_1 до y_2 и далее до y_3, при этом для каждой строки определяется x_a, z_a, x_b, z_b:

$$x_a = x_1 + (x_2 - x_1)\cdot \frac{y - y_1}{y_2 - y_1};$$

$$x_b = x_1 + (x_3 - x_1)\cdot \frac{y - y_1}{y_3 - y_1};$$

$$z_a = z_1 + (z_2 - z_1)\cdot \frac{y - y_1}{y_2 - y_1};$$

$$z_b = z_1 + (z_3 - z_1)\cdot \frac{y - y_1}{y_3 - y_1}.$$

На сканирующей строке x меняется от x_a до x_b и для каждой точки строки определяется глубина z:

$$z = z_a + (z_b - z_a)\cdot \frac{x - x_a}{x_b - x_a}$$

Реализация алгоритма вдоль сканирующей строки позволяет совместить алгоритм z-буфера с алгоритмами растровой развертки ребер и алгоритмами закраски грани.

Проиллюстрируем работу алгоритма на примере для рис. 13.12.

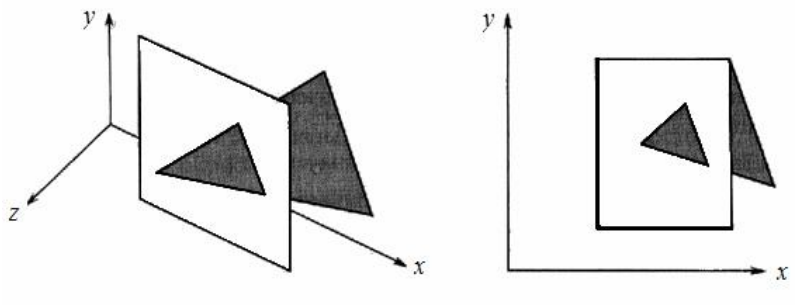


Рис. 13.12. Протыкающий треугольник

В начале в буфере кадра и в z-буфере содержатся нули. После растровой развертки прямоугольника содержимое буфера кадра будет иметь вид

0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0	0	0	1	1	1	1	1	1	1	1	1	1	1	1	1	0	0	0
0	0	0	1	1	1	1	1	1	1	1	1	1	1	1	1	0	0	0
0	0	0	1	1	1	1	1	1	1	1	1	1	1	1	1	0	0	0
0	0	0	1	1	1	1	1	1	1	1	1	1	1	1	1	0	0	0
0	0	0	1	1	1	1	1	1	1	1	1	1	1	1	1	0	0	0
0	0	0	1	1	1	1	1	1	1	1	1	1	1	1	1	0	0	0
0	0	0	1	1	1	1	1	1	1	1	1	1	1	1	1	0	0	0
0	0	0	1	1	1	1	1	1	1	1	1	1	1	1	1	0	0	0
0	0	0	1	1	1	1	1	1	1	1	1	1	1	1	1	0	0	0
0	0	0	1	1	1	1	1	1	1	1	1	1	1	1	1	0	0	0
0	0	0	1	1	1	1	1	1	1	1	1	1	1	1	1	0	0	0
0	0	0	1	1	1	1	1	1	1	1	1	1	1	1	1	0	0	0
0	0	0	1	1	1	1	1	1	1	1	1	1	1	1	1	0	0	0
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Содержимое *z*-буфера таково:

0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0	0	0	1	1	1	1	1	1	1	1	1	1	1	1	1	0	0	0
			0	0	0	0	0	0	0	0	0	0	0	0	0			
0	0	0	1	1	1	1	1	1	1	1	1	1	1	1	1	0	0	0
			0	0	0	0	0	0	0	0	0	0	0	0	0			
0	0	0	1	1	1	1	1	1	1	1	1	1	1	1	1	0	0	0
			0	0	0	0	0	0	0	0	0	0	0	0	0			
0	0	0	1	1	1	1	1	1	1	1	1	1	1	1	1	0	0	0
			0	0	0	0	0	0	0	0	0	0	0	0	0			
0	0	0	1	1	1	1	1	1	1	1	1	1	1	1	1	0	0	0
			0	0	0	0	0	0	0	0	0	0	0	0	0			
0	0	0	1	1	1	1	1	1	1	1	1	1	1	1	1	0	0	0
			0	0	0	0	0	0	0	0	0	0	0	0	0			
0	0	0	1	1	1	1	1	1	1	1	1	1	1	1	1	0	0	0
			0	0	0	0	0	0	0	0	0	0	0	0	0			
0	0	0	1	1	1	1	1	1	1	1	1	1	1	1	1	0	0	0

			0	0	0	0	0	0	0	0	0	0	0	0	0			
0	0	0	1	1	1	1	1	1	1	1	1	1	1	1	1	0	0	0
			0	0	0	0	0	0	0	0	0	0	0	0	0			
0	0	0	1	1	1	1	1	1	1	1	1	1	1	1	1	0	0	0
			0	0	0	0	0	0	0	0	0	0	0	0	0			
0	0	0	1	1	1	1	1	1	1	1	1	1	1	1	1	0	0	0
			0	0	0	0	0	0	0	0	0	0	0	0	0			
0	0	0	1	1	1	1	1	1	1	1	1	1	1	1	1	0	0	0
			0	0	0	0	0	0	0	0	0	0	0	0	0			
0	0	0	1	1	1	1	1	1	1	1	1	1	1	1	1	0	0	0
			0	0	0	0	0	0	0	0	0	0	0	0	0			
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

При обработке треугольника преобразование его в растровую форму и сравнение по глубине дает новое значение буфера кадра:

0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0	0	0	1	1	1	1	1	1	1	1	1	1	1	1	1	2	0	0
0	0	0	1	1	1	1	1	1	1	1	1	1	1	1	1	2	0	0
0	0	0	1	1	1	1	1	1	1	1	1	1	1	1	1	2	0	0
0	0	0	1	1	1	1	1	1	1	1	1	1	1	1	1	2	0	0
0	0	0	1	1	1	1	1	1	1	1	2	1	1	1	1	2	2	0
0	0	0	1	1	1	1	1	1	1	2	2	1	1	1	1	2	2	0
0	0	0	1	1	1	1	1	1	2	2	2	2	1	1	1	2	2	0
0	0	0	1	1	1	1	1	2	2	2	2	2	1	1	1	2	2	0
0	0	0	1	1	1	1	2	2	2	2	2	2	1	1	1	2	2	0
0	0	0	1	1	1	1	1	2	2	2	2	2	2	1	1	2	2	2
0	0	0	1	1	1	1	1	1	1	1	2	2	2	1	1	2	2	2
0	0	0	1	1	1	1	1	1	1	1	1	1	1	1	1	2	2	2
0	0	0	1	1	1	1	1	1	1	1	1	1	1	1	1	0	0	2
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Новое содержимое *z*-буфера таково:

0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0	0	0	1	1	1	1	1	1	1	1	1	1	1	1	1	5	0	0
			0	0	0	0	0	0	0	0	0	0	0	0	0			

```
0  0  0  1  1  1  1  1  1  1  1  1  1  1  1  1  5  0  0
         0  0  0  0  0  0  0  0  0  0  0  0  0
0  0  0  1  1  1  1  1  1  1  1  1  1  1  1  1  6  0  0
         0  0  0  0  0  0  0  0  0  0  0  0  0
0  0  0  1  1  1  1  1  1  1  1  1  1  1  1  1  6  0  0
         0  0  0  0  0  0  0  0  0  0  0  0  0
0  0  0  1  1  1  1  1  1  1  1  1  1  1  1  1  6  5  0
         0  0  0  0  0  0  0  0  1  0  0  0  0
0  0  0  1  1  1  1  1  1  1  1  1  1  1  1  1  7  5  0
         0  0  0  0  0  0  0  2  1  0  0  0  0
0  0  0  1  1  1  1  1  1  1  1  1  1  1  1  1  7  6  0
         0  0  0  0  0  0  3  2  1  1  0  0  0
0  0  0  1  1  1  1  1  1  1  1  1  1  1  1  1  7  6  0
         0  0  0  0  0  4  3  2  2  1  0  0  0
0  0  0  1  1  1  1  1  1  1  1  1  1  1  1  1  7  6  0
         0  0  0  0  5  4  3  2  2  1  0  0  0
0  0  0  1  1  1  1  1  1  1  1  1  1  1  1  1  8  7  5
         0  0  0  0  0  4  3  2  2  1  1  0  0
0  0  0  1  1  1  1  1  1  1  1  1  1  1  1  1  8  7  5
         0  0  0  0  0  0  0  0  2  1  1  0  0
0  0  0  1  1  1  1  1  1  1  1  1  1  1  1  1  8  7  6
         0  0  0  0  0  0  0  0  0  0  0  0  0
0  0  0  1  1  1  1  1  1  1  1  1  1  1  1  1  0  0  6
         0  0  0  0  0  0  0  0  0  0  0  0  0
0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0
0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0
```

13.5. Метод трассировки лучей (ray casting)

В этом методе для каждого пиксела картинной плоскости определяется ближайшая к нему грань, для чего через этот пиксел выпускается луч, находятся все его пересечения с гранями и среди них выбирается ближайшая. Алгоритм на псевдокоде можно кратко записать так:

```
for all pixels
    for all objects
        compare $z$
```

Алгоритм трассировки лучей несколько похож на алгоритм *z*-буфера, однако здесь циклы по пикселам и по объектам меняются местами.

13.6. Алгоритмы, использующие список приоритетов

При реализации всех обсуждавшихся алгоритмов удаления невидимых линий и поверхностей устанавливались приоритеты, т. е. глубины объектов сцены или их расстояния от точки наблюдения. Алгоритмы, использующие список приоритетов, пытаются получить преимущество посредством предварительной сортировки по глубине или приоритету. Цель такой сортировки состоит в том, чтобы получить окончательный список элементов сцены, упорядоченных по приоритету глубины, основанному на расстоянии от точки наблюдения. Если такой список окончателен, то никакие два элемента не будут взаимно перекрывать друг друга. Тогда можно записать все элементы в буфер кадра поочередно, начиная с элемента, наиболее удаленного от точки наблюдения. Более близкие к наблюдателю элементы будут затирать информацию о более далеких элементах в буфере кадра. Поэтому задача об удалении невидимых поверхностей решается тривиально. Эффекты прозрачности можно включить в состав алгоритма путем не полной, а частичной корректировки содержимого буфера кадра с учетом атрибутов прозрачных элементов.

Для простых элементов сцены, например для многоугольников, этот метод иногда называют алгоритмом ***художника***, поскольку он аналогичен тому способу, которым художник создает картину. Сначала художник рисует фон, затем предметы, лежащие на среднем расстоянии, и, наконец, передний план. Тем самым художник решает задачу об удалении невидимых поверхностей, или задачу видимости, путем построения картины в порядке обратного приоритета.

Для простой сцены, вроде той, что показана на рис. 13.13, a, окончательный список приоритетов можно получить непосредственно. Например, эти многоугольники можно упорядочить по их максимальному или минимальному значению координаты *z*. Однако уже для сцены, показанной на рис. 13.13 b, окончательный список приоритетов по глубине невозможно получить простой сортировкой по *z*. Если *P* и *Q* с рис. 13.13, b упорядочены по минимальному значению

координаты z (z_{min}), окажется, что P в списке приоритетов по глубине будет стоять перед Q. Если их записать в буфер кадра в таком порядке, то получится, что Q частично экранирует P. Однако фактически P частично экранирует Q. Правильный порядок в списке приоритетов будет тогда, когда P и Q поменяются местами.

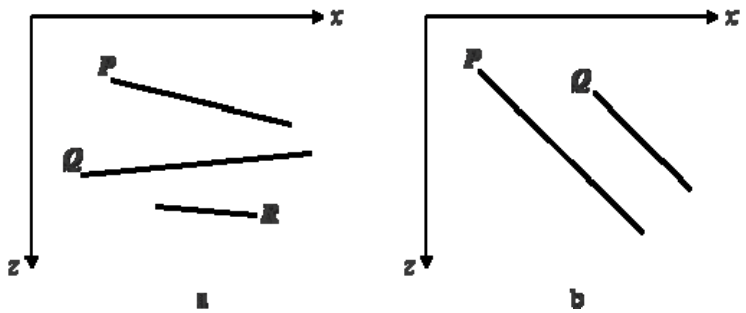


Рис. 13.13. Установление приоритетов для многоугольников

Другие трудности показаны на рис. 13.14. Здесь многоугольники циклически перекрывают друг друга. На рис. 13.14, а *P* находится впереди *Q*, который лежит впереди *R*, который, в свою очередь, находится впереди *P*. На рис. 13.14, b *P* экранирует *Q*, а *Q* экранирует *P*. Аналогичное циклическое экранирование возникает при протыкании многоугольников; например, на рис. 13.12 показано, как треугольник протыкает прямоугольник. Там прямоугольник экранируется треугольником, и наоборот. В обоих примерах окончательный список приоритетов невозможно установить сразу. Выход из положения заключается в циклическом разрезании многоугольников по линиям, образованным пересечениями их плоскостей, до тех пор, пока не будет получен окончательный список приоритетов. Такие линии показаны пунктиром на рис. 13.14.

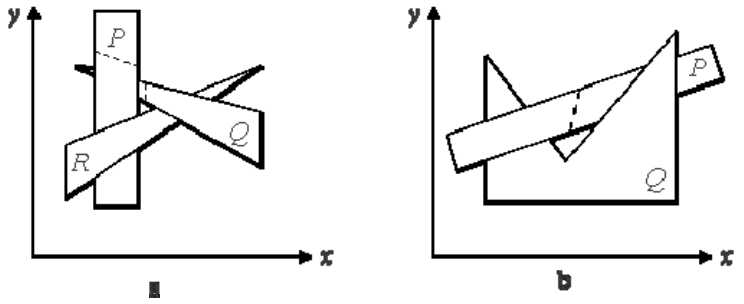


Рис. 13.14. Циклически перекрывающие многоугольники

Ньюэл М., Ньюэл Р. и Санча предложили специальный метод сортировки для разрешения конфликтов, возникающих при создании списка приоритетов по глубине. Этот метод включен в состав алгоритма Ньюэла - Ньюэла - Санча, который излагается ниже. В алгоритме динамически вычисляется новый список приоритетов перед обработкой каждого кадра сцены. Не накладывается никаких ограничений на сложность сцены и на тип многоугольников, используемых для описания элементов сцены. Первоначальный алгоритм Ньюэла - Ньюэла - Санча был предназначен для обработки трехмерных тел. Это расширение не ограничено рамками многогранников. Оно может, кроме того, обрабатывать тела смешанных типов в рамках одной сцены.

Алгоритм Ньюэла-Ньюэла-Санча для случая многоугольников

Сформировать предварительный список приоритетов по глубине, используя в качестве ключа сортировки значение z_{min} для каждого многоугольника. Первым в списке будет многоугольник с минимальным значением z_{min}. Этот многоугольник лежит дальше всех от точки наблюдения, расположенной в бесконечности на положительной полуоси z. Обозначим его через P, а следующий в списке многоугольник - через Q. Для каждого многоугольника P из списка надо проверить его отношение с Q.

Если ближайшая вершина P ($P_{z_{max}}$) будет дальше от точки наблюдения, чем самая удаленная вершина Q ($Q_{z_{min}}$), т. е. $Q_{z_{min}} \geq P_{z_{max}}$ никакая часть P не может экранировать Q. Занести P в буфер кадра (см. рис. 13.13, а).

Если $Q_{z_{min}} < P_{z_{max}}$, то P потенциальное экранирует не только Q, но также и любой другой многоугольник типа Q из списка, для которого $Q_{z_{min}} < P_{z_{max}}$. Тем самым образуется множество $\{Q\}$. Однако P может фактически и не экранировать ни один из этих многоугольников. Если последнее верно, то P можно заносить в буфер кадра. Для ответа на этот вопрос используется серия тестов, следующих по возрастанию их вычислительной сложности. Эти тесты ниже формулируются в виде вопросов. Если ответ на любой вопрос будет положительным, то P не может экранировать $\{Q\}$. Поэтому P сразу же заносится в буфер кадра. Вот эти тесты:

- Верно ли, что прямоугольные объемлющие оболочки P и Q не перекрываются по x?

- Верно ли, что прямоугольные оболочки P и Q не перекрываются по y?

- Верно ли, что P целиком лежит по ту сторону плоскости, несущей Q, которая расположена дальше от точки наблюдения (рис. 13.15, а)?

- Верно ли, что Q целиком лежит по ту сторону плоскости, несущей P, которая ближе к точке наблюдения (рис. 13.15, b)?

- Верно ли, что проекции P и Q не перекрываются?

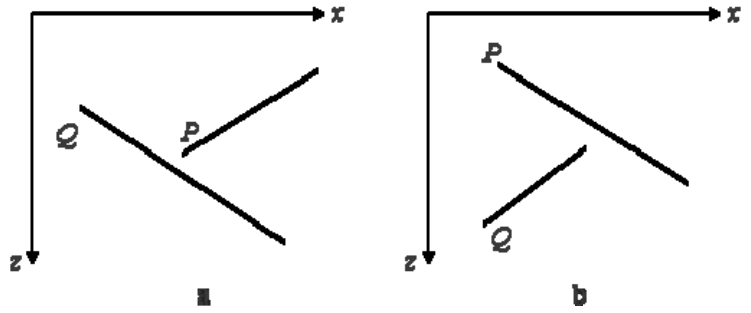


Рис. 13.15. Тесты для перекрывающихся многоугольников
Каждый из этих тестов применяется к каждому элементу {*Q*}. Если ни один из них не дает положительного ответа и не заносит *P* в буфер кадра, то *P* может закрывать *Q*.

Поменять P и Q местами, пометив позицию Q в списке. Повторить тесты с новым списком. Это дает положительный результат для сцены с рис. 13.13, b.

Если сделана попытка вновь переставить Q, значит, обнаружена ситуация циклического экранирования (см. рис. 13.14.). В этом случае P разрезается плоскостью, несущей Q, исходный многоугольник P удаляется из списка, а его части заносятся в список. Затем тесты повторяются для нового списка. Этот шаг предотвращает зацикливание алгоритма.

13.7. Алгоритм Варнока (Warnock)

Алгоритм Варнока является одним из примеров алгоритма, основанного на разбиении картинной плоскости на части, для каждой из которых исходная задача может быть решена достаточно просто.

Поскольку алгоритм Варнока нацелен на обработку картинки, он работает в пространстве изображения. В пространстве изображения рассматривается окно и решается вопрос о том, пусто ли оно, или его содержимое достаточно просто для визуализации. Если это не так, то окно разбивается на фрагменты до тех пор, пока содержимое фрагмента не станет достаточно простым для визуализации или его размер не достигнет требуемого предела разрешения.

Кратко опишем оригинальную версию алгоритма, предложенного Варноком.

Разобьем видимую часть картинной плоскости на четыре равные части и рассмотрим, каким образом могут соотноситься между собой проекции граней получившейся части картинной плоскости.

Возможны четыре различных случая:

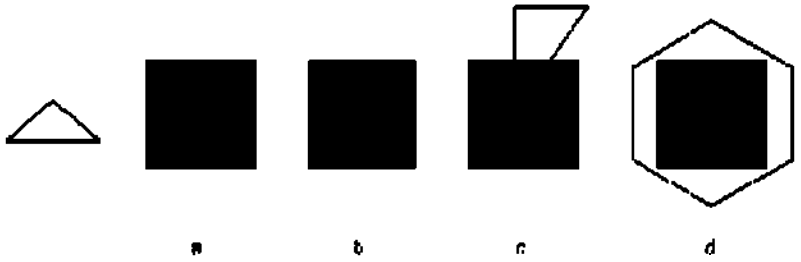


Рис. 13.16. Соотношение проекции грани с подокном

1. Проекция грани полностью накрывает область (рис 13.16, d);

2. Проекция грани пересекает область, но не содержится в ней полностью (рис. 13.16, c);

3. Проекция грани целиком содержится внутри области (рис. 13.16, b);

4. Проекция грани не имеет общих внутренних точек с рассматриваемой областью (рис 13.16, a).

Очевидно, что в последнем случае грань вообще никак не влияет на то, что видно в данной области.

Сравнивая область с проекциями всех граней, можно выделить случаи, когда изображение, получающееся в рассматриваемой области, определяется сразу:

♦ Проекция ни одной грани не попадает в область.

♦ Проекция только одной грани содержится в области или пересекает область. В этом случае проекции грани разбивают всю область на две части, одна из которых соответствует этой проекции.

♦ Существует грань, проекция которой полностью накрывает данную область, и эта грань расположена к картинной плоскости ближе, чем все остальные грани, проекции которых пересекают данную область. В данном случае область соответствует этой грани.

Если ни один из рассмотренных трех случаев не имеет места, то снова разбиваем область на четыре равные части и проверяем выполнение этих условий для каждой из частей. Те части, для которых таким образом не удалось установить видимость, разбиваем снова и т. д.

Естественно, возникает вопрос о критерии, на основании которого прекращать разбиение. В качестве очевидного критерия можно взять размер области. Как только размер области станет не больше размера одного пиксела, то производить дальнейшее разбиение не имеет смысла и для данной области ближайшая к ней грань определяется явно, как в методе трассировки лучей.

Иногда, для устранения лестничного эффекта, процесс разбиения проводится до размеров, меньших, чем разрешение экрана, на один пиксель. При этом усредняются атрибуты подпикселей, чтобы определить атрибуты самих пикселей.

При помощи изложенного алгоритма можно удалить либо невидимые линии, либо невидимые поверхности. Однако простота критерия разбиения, а также негибкость способа разбиения приводят к тому, что количество подразбиений оказывается велико. Можно повысить эффективность этого алгоритма, усложнив способ и критерий разбиения. На рис. 13.17, a показан другой общий способ разбиения и дано его сравнение с изложенным ранее жестким способом, представленным на рис. 13.17, b.

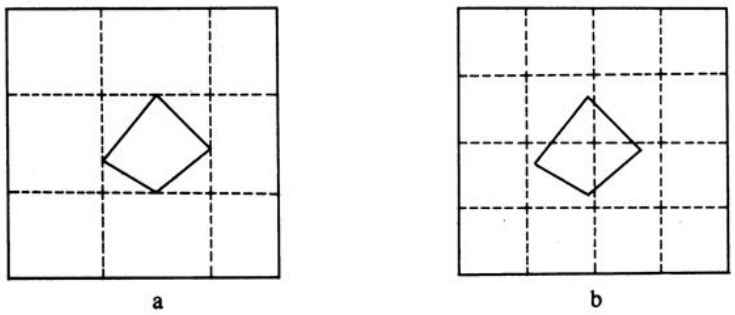


Рис. 13.17. Способы разбиения окна

Разбиение, показанное на рис. 13.17, а, получается с использованием прямоугольной объемлющей оболочки многоугольника. Заметим, что оболочки при этом могут быть неквадратными. Этот способ можно рекурсивно применить к любому многоугольнику, который полностью охвачен каким-нибудь окном или оболочкой. Если в окне содержится только один многоугольник и если он целиком охвачен этим окном, то его легко изобразить, не проводя дальнейшего разбиения. Такой способ разбиения полезен, в частности, при минимизации числа разбиений для простых сцен. Однако с ростом сложности сцены его преимущество сходит на нет.

13.8. Алгоритм Вейлера-Азертона (Weiler-Atherton)

Разбиение картинной плоскости можно производить не только прямыми, параллельными координатным осям, но и по границам проекций граней. В результате получается точное решение задачи.

Предлагаемый метод работает с проекциями граней на картинную плоскость.

В качестве первого шага производится сортировка всех граней по глубине (front-to-back).

Затем из списка оставшихся граней берется ближайшая грань A и все остальные грани обрезаются по этой грани. Если проекция грани B пересекает проекцию грани A, то грань B разбивается на части так, что каждая часть либо содержится в грани A, либо не имеет с ней общих внутренних точек.

Таким образом, получаются два множества граней: F_{in} – грани, проекции которых содержатся в проекции грани A (сюда входит и сама грань A), и F_{out} – грани, проекции которых не имеют общих внутренних точек с проекцией грани A.

Множество F_{in} обычно называют множеством граней, внутренних по отношению к A.

Однако во множестве F_{in} могут быть грани, лежащие к наблюдателю ближе, чем сама грань A (это возможно, например, при циклическом наложении граней). В этом случае каждая такая грань используется для разбиения всех граней из множества F_{in} (включая исходную грань A). Когда рекурсивное разбиение завершится, то все грани из первого множества выводятся из набора оставшихся граней (их уже ничто не может закрывать). Затем из набора оставшихся граней F_{out} берется очередная грань и процедура повторяется.

Рассмотрим простейший случай для двух граней A и B (рис. 13.18).

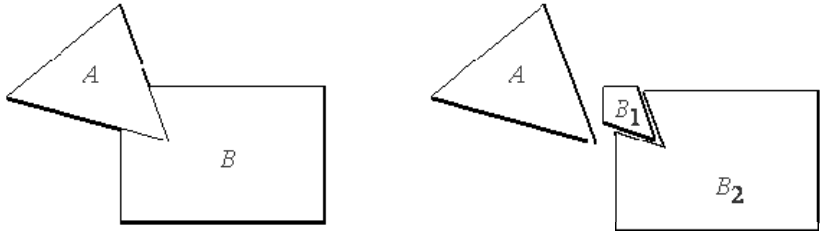


Рис. 13.18. Пример разбиения граней

Будем считать, что грань A расположена ближе, чем грань B. Тогда на первом шаге для разбиения используется именно грань A. Грань B

разбивается на две части. Часть B_1 попадает в первое множество F_{in} и, так как она лежит дальше грани A, удаляется.

После этого выводится грань A и в списке оставшихся граней остается только грань B_2. Так как кроме нее других граней не осталось, то эта грань выводится, и на этом работа завершается.

14. Цвет как характеристика восприятия объекта искусственным интеллектом

14.1.Что такое цветовая модель?

Многим приходилось слышать такие словосочетния, как "световая модель", "файл для печати должен быть в CMYK, а для размещения на сайте - в RGB". Кто-то, может быть, даже знает о существовании таких цветовых моделей, как GreyScale, LAB, HSB и HLS. А что же такое эти "Цветовые модели? Чем цветовая модель CMYK отличается от RGB или LAB?

Живем мы на белом свете. А свет этот можно разделить на множество разных оттенков. Насколько мы знаем, первым, кому в голову пришла эта глупая мысль, был Исаак Ньютон. Он делил свет через призму на семь основных цветов: красный, оранжевый, желтый, зеленый, голубой, синий и фиолетовый.

Цветовые характеристики несут информацию об отражательных свойствах. Различие отражательной способности объекта в разных участках спектрального диапазона обеспечивает возможность извлечения важной биологической информации об объекте. Ведь именно по цвету можно оценить, например, созрел ли плод, поражены ли заболеванием сельскохозяйственные культуры и многое другое. Теория цветового зрения и сегодня находится в незавершенной стадии развития. Проблема состоит в том, что **многочисленные модели описывают цветовое зрение, но не являются теорией цветового зрения**, поскольку ни одной из них не удается строго ответить на все вопросы об установленных фактах, относящихся к психофизическому и физиологическому аспектам цветового зрения. Начиная с опытов Ньютона и Максвелла, было предложено множество

теорий, описывающих цветовое зрение человека. В классической трехцветной модели цветового зрения, разработанной Томасом Юнгом в 1802 году, предполагается, что существуют три компонента любого цветоощущения, которые являются **аддитивными** основными **цветами**: это красный *(R),* зеленый (G) и синий *(B).* На самом деле имеется бесконечное множество основных цветов, но чтобы получить максимальный диапазон смешанных цветов, следует пользоваться *RGB.* Единственное условие правильного выбора основных цветов состоит в том, что при смешении двух из них мы не должны получать третий цвет. Юнг постулировал, что поскольку трехкомпонентность цвета не имеет обоснования в теории света, то цвет является свойством самого глаза. Глаз анализирует каждый цвет в отдельности и передает сигналы о нем в мозг по трем типам нервных волокон: один тип передает сигнал о наличии *R,* второй - *G,* третий - *B.* На 50 лет теория Юнга была отвергнута и предана забвению. В 1852 году к ней одновременно обратились немецкий физик и физиолог Герман фон Гельмгольц и шотландский физик Джеймс Клерк Максвелл. Гельмгольц при попытке получить сине-зеленый цвет с длиной волны 500 нм смешением *BG,* заметил, что его нельзя получить путем аддитивного сложения трех основных цветов. Смесь получается белесая, менее насыщенная по сравнению со спектральным цветом. Через 10 лет Гельмгольц понял, что результаты опытов можно объяснить и на основе трех основных механизмов, исходя из предположения о том, что они обладают спектральной чувствительностью в широком, частично перекрывающемся диапазоне. При таком подходе, даже если раздражитель чистый в оптическом смысле, ответная реакция глаза таковой не является. Максвелл одним из первых признал теорию Юнга и **занялся разработкой точных методов измерения цветов**. Он использовал цветовой треугольник Юнга, поместив основные цвета *RGB* в вершины равностороннего треугольника. Результирующий цвет любой смеси *RGB* располагается в центре тяжести трех масс. Результирующая аддитивной смеси двух цветов находится в их центре тяжести и поэтому лежит на прямой, соединяющей эти цвета. Этот **закон центра тяжести является свойством всех плоских цветовых диаграмм**. Теория Юнга - Гельмгольца не соответствовала цветовым ощущениям. Человек в состоянии различать, по меньшей мере, четыре качественно разных цветовых ощущения: красного, желтого, зеленого и синего цветов, - если к ним добавить белый, то получится пять.
В 1870 году немецкий физиолог Эвальд Геринг сформулировал оппонентную теорию цветового зрения. Он опирался на существование пяти психологических ощущений и считал, что они действуют в

противоположных парах. В парах: красный и зеленый, желтый и синий -цвета являются противоположными и не смешиваются.

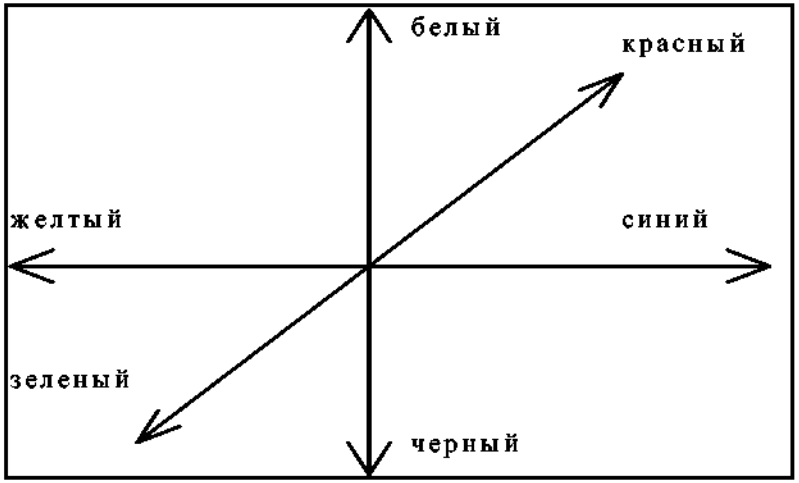


Рис. 14.1 Оси противоположных цветов в соответствии с оппонентной теорией Геринга.

После Геринга был столетний перерыв в развитии теории цветового зрения. В 1953 году Томсон и Райт опубликовали кривые спектральной чувствительности к красному, синему и зеленому диапазонам спектра. В 1964 году две группы американских ученых (Маркс, Добелл, Мак - Никол в опытах на сетчатке серебряного карася, обезьяны и человека, и Браун и Уолд на сетчатке человека) обнаружили **три типа колбочек, поглощающих свет в различных частях спектра**. Согласно современным данным на рецептурном уровне свет регистрируется тремя различными типами колбочек (как постулировано в теории Юнга - Гельмгольца), и эти рецепторы обладают чувствительностью к *R, G, B* - частям спектра. Однако, поступающая от них **информация**, **повидимому, преобразуется в импульсные разряды и до передачи в мозг кодируется в сетчатке. Эта закодированная информация посылается в виде сигнала о яркости из всех трех типов колбочек, а также в виде разностных сигналов каждых двух цветов. Подключается и второй яркостный сигнал, берущий начало, вероятно, от независимой палочковой системы. Мозг воспринимает закодированную информацию о яркости и разностные цветовые сигналы. Таков механизм цветного зрения в соответствии с зонной теорией Адамса** .

Концепция построения систем цветного телевидения основана на принципе постоянной яркости и согласуется с зонной теорией Адамса. В рамках трехкомпонентной теории цвета набор основных цветов можно выбрать многими способами, этим объясняется большое количество координатных систем, предложенных для количественного описания цвета. Описание этих координатных систем приводится как в фундаментальных исследованиях по цвету, так и в научно-технических статьях при описании различных алгоритмов обработки. Методы анализа цветных изображений зависят от цветового координатного пространства, выбор цветового координатного пространства определяет эффективность метода.

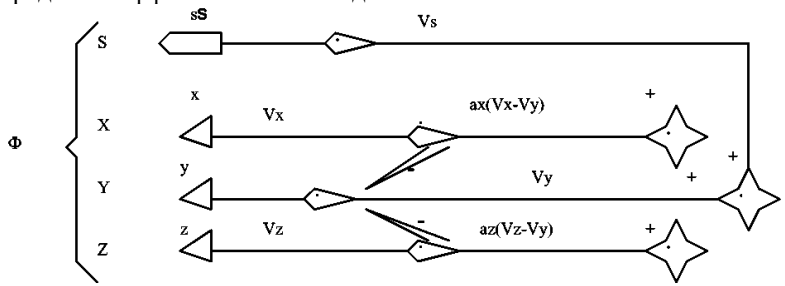
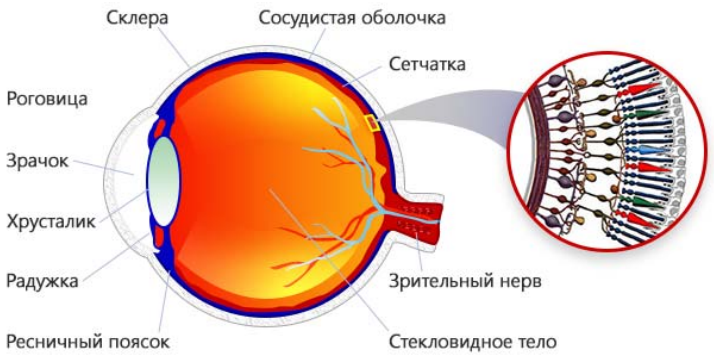


Рис. 14.2 Механизм цветного зрения в соответствии с зонной теорией Адамса.

Максвелл развивал идеи Томаса Юнга, который выдвинул как мы говорили, предположение о существовании трех основных цветов: красного, зеленого и синего — в соответствии с тремя типами чувствительных волокон в сетчатке глаза. В сетчатке есть два вида фоторецепторов: палочки и колбочки. Колбочки отвечают за цветовое зрение и, в свою очередь, делятся еще на три вида: одни чувствительны к красно-желтой, другие — к зелено-желтой, а третьи — к сине-фиолетовой части спектра.

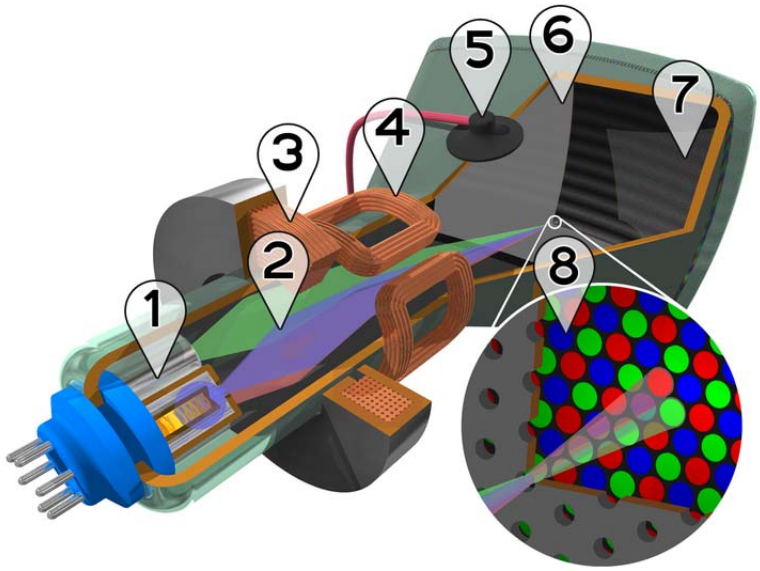


Максвелл с помощью своего волчка продемонстрировал, что белый цвет нельзя получить смешением синего, красного и желтого, как считалось ранее, а основными цветами являются красный, зеленый и синий.

Как монитор передает цвета

Хотя Максвелл проводил свои исследования еще в XIX веке, цветовая модель RGB на практике стала использоваться позже — когда появились телевизоры и мониторы, сначала с электронно-лучевой трубкой, а потом жидкокристаллические и плазменные.

В ЭЛТ изображение создается с помощью трех электронных прожекторов, каждый из которых излучает свет своего цвета. На экран нанесен люминофор — вещество, которое светится под воздействием этих прожекторов. Причем люминофор тоже трех видов: один светится от излучения красной пушки, второй — от зеленой, третий — от синей.



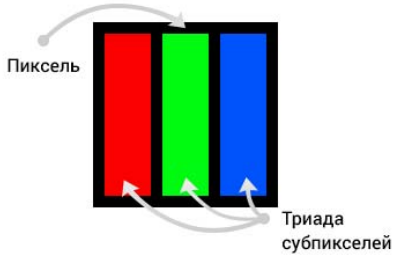
Устройство цветного кинескопа:

1. Электронные пушки
2. Электронные лучи
3. Фокусирующая катушка
4. Отклоняющие катушки
5. Анод
6. Маска, благодаря которой красный луч попадает на красный люминофор, зеленый луч — на зеленый люминофор, синий — на синий.
7. Красные, зелёные и синие зёрна люминофора
8. Маска и зёрна люминофора (увеличено)

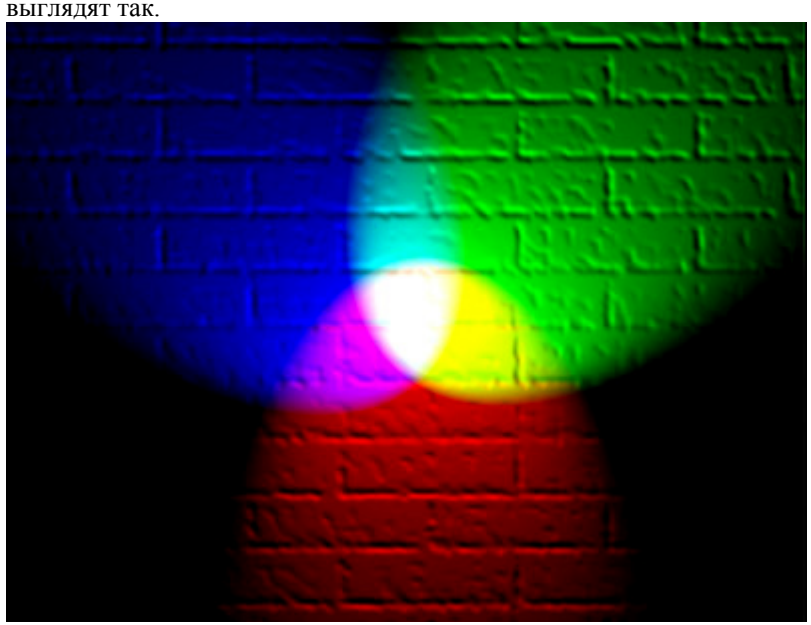
При всех конструктивных и технологических отличиях от ЭЛТ, жидкокристаллические и плазменные мониторы работают по тому же принципу: под воздействием энергии загорается красный, зеленый или синий люминофор.

Минимальная единица изображения, создаваемого монитором, называется *пикселем*. Цвет пикселя получается из комбинации

входящих в него трех точек люминофора (эти три точки называются триадами).



Такую картинку можно увидеть, посмотрев на монитор в лупу. Пиксели не обязательно бывают прямоугольные, но чаще всего они выглядят так.



14.2. Методы закраски

14.2.1. Диффузное отражение и рассеянный свет

Матовые поверхности обладают свойством ***диффузного отражения***, т. е. равномерного по всем направлениям рассеивания света. Поэтому кажется, что поверхности имеют одинаковую яркость независимо от угла обзора. Для таких поверхностей справедлив закон косинусов Ламберта, устанавливающий соответствие между количеством отраженного света и косинусом угла θ между направлением $\overline{L}$ на точечный источник света интенсивности I_p и нормалью $\overline{N}$ к поверхности (рис. 14.1). При этом количество отраженного света не зависит от положения наблюдателя.

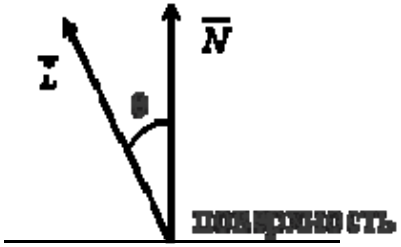


Рис. 14.1. Падающий свет и нормаль к поверхности

Освещенность рассеянным светом вычисляется по формуле

$$I_d = I_p \cdot k_d \cdot \cos\theta.$$

Значение коэффициента диффузного отражения k_d является константой в диапазоне (0, 1) и зависит от материала. Если векторы $\overline{L}$ и $\overline{N}$ нормированы, то, используя скалярное произведение, формулу освещенности можно записать так:

$$I_d = I_p \cdot k_d \cdot (\overline{L} \cdot \overline{N}).$$

Предметы, освещенные одним точечным источником света, выглядят контрастными. Этот эффект аналогичен тому, который можно наблюдать, когда предмет, помещенный в темную комнату, виден при свете направленной на него фотовспышки. В данной ситуации, в отличие, от большинства реальных визуальных сцен, отсутствует рассеянный свет, под которым здесь понимается свет постоянной яркости, созданный многочисленными отражениями от различных поверхностей. Такой свет практически всегда присутствует в реальной обстановке. Даже если предмет защищен от прямых лучей, исходящих от точечного источника света, он все равно будет виден из-за наличия рассеянного света. Учитывая это, формулу окраски можно записать так:

$$I_d = I_a \cdot k_a + I_p \cdot k_d \cdot (\overline{L} \cdot \overline{N})$$

Рассеянный свет представлен членом I_a, и k_a определяет количество рассеянного света, которое отражается от поверхности предмета.

Точечный источник света удобнее всего расположить в позиции, совпадающей с глазом наблюдателя. Тени в этом случае отсутствуют, а лучи света, падающие на поверхность, окажутся параллельными. Однако теперь, если две поверхности одного цвета параллельны друг другу и их изображения перекрываются, нормали к поверхностям совпадают и, следовательно, поверхности закрашиваются одинаково, различить их невозможно. Этот эффект можно устранить, если учесть, что энергия падающего света убывает пропорционально квадрату расстояния, которое свет проходит от источника до поверхности и обратно к глазу наблюдателя. Обозначая это расстояние за R, запишем:

$$I_d = I_a \cdot k_a + I_p \cdot k_d \cdot (\overline{L} \cdot \overline{N})/R^2.$$

Однако данным правилом на практике трудно воспользоваться. Для параллельной проекции, когда источник света находится в бесконечности, расстояние R также становится бесконечным. Даже в

случае центральной проекции величина $1/R^2$ может принимать значения в широком диапазоне, поскольку точка зрения часто оказывается достаточно близкой к предмету. В результате закраска поверхностей, которые имеют одинаковые углы θ между $\overline{N}$ и $\overline{L}$, будет существенно различаться. Большей реалистичности можно достичь, если заменить R^2 на $r+k$, где k – некоторая константа, а r – расстояние от центра проекции до поверхности:

$$I_d = I_a \cdot k_a + I_p \cdot k_d \cdot (\overline{L} \cdot \overline{N})/(r+k).$$

Для представления диффузного отражения от цветных поверхностей уравнения записываются отдельно для основных цветов модели CMY (голубого, пурпурного и желтого). При этом константы отражения для этих цветов задаются тройкой чисел (k_{dc}, k_{dm} k_{dy}). Эти цвета используются, поскольку отражение света является субтрактивным процессом. Поэтому интенсивность для цветного изображения описывается тремя уравнениями:

$$I_{dc} = I_{ac} \cdot k_{ac} + I_{pc} \cdot k_{dc} \cdot (\overline{L} \cdot \overline{N})/(r+k)$$ (для голубой компоненты);

$$I_{dm} = I_{am} \cdot k_{am} + I_{pm} \cdot k_{dm} \cdot (\overline{L} \cdot \overline{N})/(r+k)$$ (для пурпурной

$$I_{dy} = I_{ay} \cdot k_{ay} + I_{py} \cdot k_{dy} \cdot (\overline{L} \cdot \overline{N})/(r+k)$$ (для желтой компоненты).

14.2.2. Зеркальное отражение

Зеркальное отражение можно получить от любой блестящей поверхности. Осветите ярким светом яблоко – световой блик на яблоке возникает в результате зеркального отражения, а свет, отраженный от остальной части, появится в результате диффузного отражения. Отметим также, что в том месте, где находится световой блик, яблоко кажется не красным, а скорее белым, т. е. окрашенным в цвет падающего цвета.

Если мы изменим положение головы, то заметим, что световой блик тоже сместится. Это объясняется тем, что блестящие поверхности отражают свет неодинаково по всем направлениям. От идеального

зеркала свет отражается только в том направлении, для которого углы падения и отражения совпадают. Это означает, что наблюдатель сможет увидеть зеркально отраженный свет только в том случае, если угол α (рис. 14.2.) равен нулю.

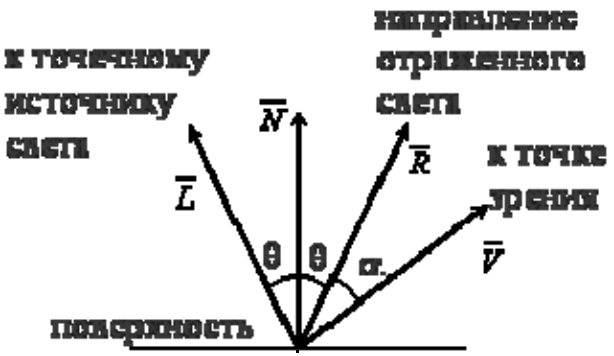


Рис. 14.2. Зеркальное отражение

Для неидеальных отражающих поверхностей, таких, как яблоко, интенсивность отраженного света резко падает с ростом α. В модели предложенной Фонгом, быстрое убывание интенсивности описывается функцией $\cos^n \alpha$, где *n* обычно лежит в диапазоне 1–200, в зависимости от вида поверхности. Для идеального отражателя *n* бесконечно велико. В основе такой модели лежит эмпирическое наблюдение, а не фундаментальное понимание процесса зеркального отражения.

Количество падающего света, которое зеркально отражается в случае реальных материалов, зависит от угла падения θ. Обозначим зеркально отражаемую долю света через $W(\theta)$, тогда

$$I_d = I_a \cdot k_a + \frac{I_l}{r+k} \cdot [k_d \cdot \cos\theta + W(\theta) \cdot \cos^n \alpha].$$

Если векторы направления отраженного света и направления к точке зрения $\overline{R}$ и $\overline{V}$ нормированы, то $\cos\alpha = (\overline{R} \cdot \overline{V})$. Часто в качестве

$W(\theta)$ служит константа k_s, которая выбирается таким образом, чтобы получающиеся результаты были приемлемы с эстетической точки зрения. В этом случае уравнение с учетом зеркального отражения можно записать так:

$$I_d = I_a \cdot k_a + \frac{I_l}{r+k} \cdot [k_d \cdot (\overline{L} \cdot \overline{N}) + k_s \cdot (\overline{R} \cdot \overline{V})^n].$$

Для цветного изображения описываются три уравнения: для голубого, пурпурного и желтого цветов:

$$I_{dc} = I_{ac} \cdot k_{ac} + \frac{I_{pc}}{r+k} \cdot [k_{dc} \cdot (\overline{L} \cdot \overline{N}) + k_s \cdot (\overline{R} \cdot \overline{V})^n];$$

$$I_{dm} = I_{am} \cdot k_{am} + \frac{I_{pm}}{r+k} \cdot [k_{dm} \cdot (\overline{L} \cdot \overline{N}) + k_s \cdot (\overline{R} \cdot \overline{V})^n];$$

$$I_{dy} = I_{ay} \cdot k_{ay} + \frac{I_{py}}{r+k} \cdot [k_{dy} \cdot (\overline{L} \cdot \overline{N}) + k_s \cdot (\overline{R} \cdot \overline{V})^n].$$

Если источник света расположен в бесконечности, для заданного многоугольника произведение $(\overline{L} \cdot \overline{N})$ является константой, а $(\overline{R} \cdot \overline{V})$ меняет значение в многоугольнике.

Кроме эмпирической модели Фонга, для зеркального отражения разработана модель Торрэнса-Спэрроу, которая представляет собой теоретическую обоснованную модель отражающей поверхности. В этой модели предполагается, что поверхность является совокупностью микроскопических граней, каждая из которых – идеальный отражатель. Ориентация любой грани задается функцией распределения вероятностей Гаусса.

14.2.3. Однотонная закраска полигональной сетки

Существует три основных способа закраски объектов, заданных полигональными сетками. В порядке возрастания сложности ими являются:

1) однотонная закраска;

2) метод Гуро (основан на интерполяции значений интенсивности);

3) метод Фонга (основан на интерполяции векторов нормали).

В каждом из этих случаев может быть использована любая из моделей закраски, описанная выше.

При однотонной закраске вычисляют один уровень интенсивности, который используется для закраски всего многоугольника. При этом предполагается, что:

1. Источник света расположен в бесконечности, поэтому произведение ($\overline{L} \cdot \overline{N}$) постоянно на всей полигональной грани.
2. Наблюдатель находится в бесконечности, поэтому произведение ($\overline{N} \cdot \overline{V}$) постоянно на всей полигональной грани.
3. Многоугольник представляет реальную моделируемую поверхность, а не является аппроксимацией криволинейной поверхности. Если какое-либо из первых двух предположений оказывается неприемлемым, можно воспользоваться усредненными значениями $\overline{L}$ и $\overline{V}$, вычисленными, например, в центре многоугольника.

Последнее предположение в большинстве случаев не выполняется, но оказывает существенно большее влияние на получаемое изображение, чем два других. Влияние состоит в том, что каждая из видимых полигональных граней аппроксимированной поверхности хорошо отличима от других, поскольку интенсивность каждой из этих граней

отличается от интенсивности соседних граней. Различие в окраске соседних граней хорошо заметно вследствие эффекта полос Маха.

14.2.4. Метод Гуро

Метод закраски, который основан на интерполяции интенсивности и известен как метод Гуро (по имени его разработчика), позволяет устранить дискретность изменения интенсивности. Процесс закраски по методу Гуро осуществляется в четыре этапа:

1. Вычисляются нормали ко всем полигонам.

2. Определяются нормали в вершинах путем усреднения нормалей по всем полигональным граням, которым принадлежит вершина (рис 14.3).

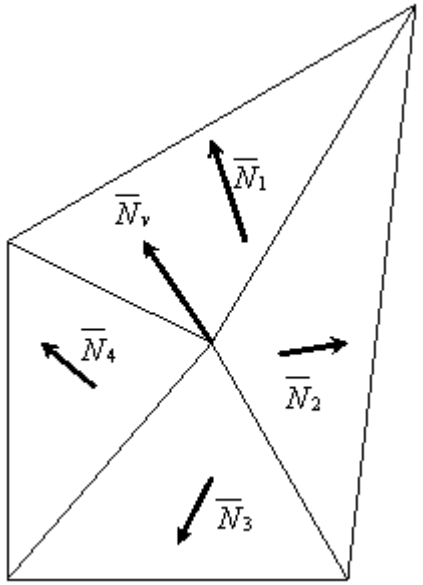


Рис. 14.3. Нормали к вершинам:
$\overline{N}_v = (\overline{N}_1 + \overline{N}_2 + \overline{N}_3 + \overline{N}_4 + \overline{N}_v)/4$

3. Используя нормали в вершинах и применяя произвольный метод закраски, вычисляются значения интенсивности в вершинах.

4. Каждый многоугольник закрашивается путем линейной интерполяции значений интенсивностей в вершинах сначала вдоль каждого ребра, а затем и между ребрами вдоль каждой сканирующей строки (рис. 14.4).

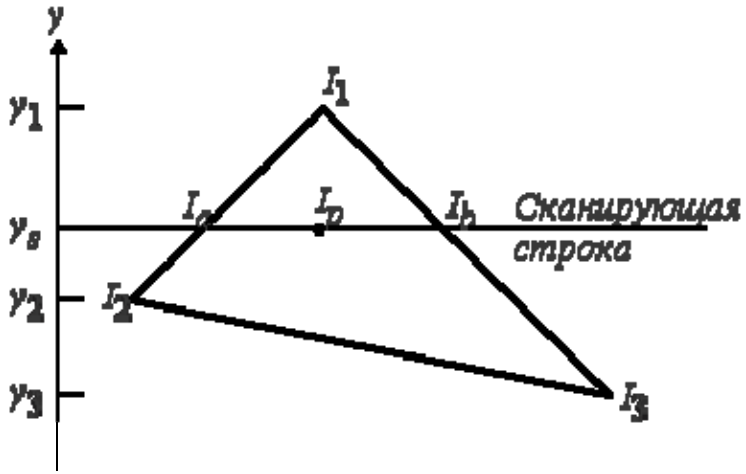


Рис. 14.4. Интерполяция интенсивностей

Интерполяция вдоль ребер легко объединяется с алгоритмом удаления скрытых поверхностей, построенным на принципе построчного сканирования. Для всех ребер запоминается начальная интенсивность, а также изменение интенсивности при каждом единичном шаге по координате *y*, Заполнение видимого интервала на сканирующей строке производится путем интерполяции между значениями интенсивности на двух ребрах, ограничивающих интервал (рис 14.4).

$$I_a = I_1 \frac{y_1 - y_2}{y_1 - y_2} + I_2 \frac{y_1 - y_s}{y_1 - y_2};$$

$$I_b = I_1 \frac{y_1 - y_3}{y_1 - y_3} + I_3 \frac{y_1 - y_1}{y_1 - y_3};$$

$$I_p = I_a \frac{x_b - x_p}{x_b - x_a} + I_b \frac{x_p - x_a}{x_b - x_a}.$$

Для цветных объектов отдельно интерполируется каждая из компонент цвета.

14.2.5. Метод Фонга

В методе закраски, разработанном Фонгом, используется интерполяция вектора нормали N к поверхности вдоль видимого интервала на сканирующей строке внутри многоугольника, а не интерполяция интенсивности. Интерполяция выполняется между начальной и конечной нормалями, которые сами тоже являются результатами интерполяции вдоль ребер многоугольника между нормалями в вершинах. Нормали в вершинах, в свою очередь, вычисляются так же, как в методе закраски, построенном на основе интерполяции интенсивности.

В каждом пикселе вдоль сканирующей строки новое значение интенсивности вычисляется с помощью любой модели закраски. Заметные улучшения по сравнению с интерполяцией интенсивности наблюдаются в случае использования модели с учетом зеркального отражения, так как при этом более точно воспроизводятся световые блики. Однако даже если зеркальное отражение не используется, интерполяция векторов нормали приводит к более качественным результатам, чем интерполяция интенсивности, поскольку аппроксимация нормали в этом случае осуществляется в каждой точке. При этом значительно возрастают вычислительные затраты.

Чтобы закрасить куски бикубической поверхности, для каждого пиксела, исходя из уравнений поверхности, вычисляется нормаль к поверхности. Этот процесс тоже достаточно дорогой. Затем с помощью любой модели закраски определяется значение интенсивности. Однако прежде чем применить метод закраски к плоским или бикубическим поверхностям, необходимо иметь информацию о том, какие источники света (если они имеются) в действительности освещают точку. Поэтому мы должны рассматривать также и тени.

14.2.6. Тени

Алгоритмы затенения в случае точечных источников света идентичны алгоритмам удаления скрытых поверхностей. В алгоритме удаления скрытых поверхностей определяются поверхности, которые можно увидеть из точки зрения, а в алгоритме затенения выделяются поверхности, которые можно «увидеть» из источника света. Поверхности, видимые как из точки зрения, так и из источника света, не лежат в тени. Те же поверхности, которые видимы из точки зрения, но невидимы из источника света, находятся в тени. Эти рассуждения можно легко распространить на случай нескольких источников света. Отметим, однако, что, используя такой простой подход, нельзя смоделировать тени от распределенных источников света. При наличии таких источников потребуется вычислять как тени, так и полутени.

Поскольку алгоритмы затенения и удаления скрытых поверхностей одинаковы, представляется возможным обрабатывать описание объекта, используя лишь один из этих алгоритмов, последовательно применяя его к точке зрения и к каждому из точечных источников света. Совокупность полученных результатов позволяет определить, какие части объекта видимы наблюдателю и какие видны из одного или нескольких источников света. На основании этой информации осуществляется закраска сцены. Если правильно организовать вычислительный процесс, определение теней можно проводить лишь один раз для серии сцен, которые состоят из одних и тех же объектов, рассматриваемых с различных точек зрения. Источники света предполагаются неподвижными относительно объектов. Все это оказывается возможным потому, что тени не зависят от положения точки зрения.

14.2.7. Поверхности, пропускающие свет

Поверхности могут обладать не только свойствами зеркального и диффузного отражения, но и свойствами направленного и диффузного пропускания. Направленное пропускание света происходит сквозь прозрачные вещества (например, стекло или отшлифованный люциит). Через них обычно хорошо видны предметы, даже несмотря на то, что лучи света, как правило, преломляются, т. е. отклоняются от первоначального направления. Диффузное пропускание света происходит сквозь просвечивающие материалы (например, замерзшее

стекло), в которых поверхностные или внутренние неоднородности приводят к беспорядочному перемешиванию световых лучей. Поэтому когда предмет рассматривается через просвечивающее вещество, его очертания размыты.

Изложим кратко идею только одного метода (подход Уиттеда), учитывающего пропускающий свет. Данный подход основан на использовании алгоритмов трассировки лучей. Его основная идея заключается в трассировании световых лучей и определении, какие из этих лучей попадают в точку зрения. К сожалению, из каждой точки источника света исходит бесконечное число лучей, причем большинство из них никогда не достигает точки зрения. Поэтому трассирование начинается из точки зрения и лучи отслеживаются в обратном направлении через каждый пиксел к их источнику. Луч света, падающий на поверхность, в общем случае разделяется на три части: диффузно отраженный свет, зеркально отраженный свет и пропущенный (и, следовательно, преломленный) свет. Аналогично луч света, исходящий от поверхности объекта, в общем случае является суммой составляющих от трех источников. Это означает, что каждый раз, когда луч исходит от объекта, возможно появление трех новых лучей, которые должны быть оттрассированы. К сожалению, диффузное отражение приводит к появлению бесконечного числа лучей, поэтому трассируются только лучи, появляющиеся в результате зеркального отражения и преломления. Для моделирования рассеянного и диффузного света используется уравнение

$$I_d = I_a \cdot k_a + I_p \cdot k_d \cdot (\overline{L} \cdot \overline{N})$$

14.2.8. Детализация поверхностей

Существуют два способа детализации поверхности: цветом и фактурой. В результате применения к гладкой поверхности детализации цветом форма поверхности не изменяется, если же производится детализация фактурой – поверхность становится шероховатой.

14.2.8.1. Детализация цветом

Детализацию цветом на глубоком уровне легко осуществить путем введения многоугольников детализации поверхности, чтобы выделить

особенности (такие, как двери, окна и надписи) на основном многоугольнике (например, на стене здания). Многоугольники детализации поверхности лежат в одной плоскости с основными многоугольниками и так помечены в структуре данных, чтобы алгоритм удаления скрытых поверхностей мог присвоить им более высокие приоритеты, чем основным многоугольникам.

По мере того как детализация цветом становится более тонкой и сложной, непосредственное моделирование при помощи многоугольников становится менее практичным.

14.2.8.2. Детализация фактурой

Идея детализации фактурой состоит в отображении массива узора, представляющего собой оцифрованное изображение, на плоскую или криволинейную поверхность. Значения из массива узора используются для масштабирования диффузной компоненты интенсивности.

Один пиксел на экране может покрывать несколько элементов массива узора. Чтобы избежать проблем, связанных с лестничным эффектом, необходимо учитывать все затрагивающие пиксел элементы. Для этого определяются четыре точки в массиве узора, которые соответствуют четырем углам пиксела. Эти точки в массиве узора образуют четырехугольник. Значения попадающих в него элементов взвешиваются с учетом доли каждого элемента, а затем суммируются.

Отображение при такой детализации проводится в два этапа:

1. Фиксированное отображение рисунка на поверхность объекта.

2. Видовое преобразование объекта на экран.

Отображение массива узора влияет на расцветку поверхности, однако поверхность продолжает казаться геометрически гладкой. Существует два способа нанесения на поверхность ***деталей фактуры***. В первом из них непосредственное геометрическое моделирование фактуры не производится, и тем не менее получается хороший визуальный эффект. Для этого вносится возмущение в нормаль к поверхности до ее

использования в модели закраски. Эти возмущения моделируют небольшие неровности на поверхности.

Второй способ основывается на использовании фрактальных поверхностей, т. е. класса нерегулярных форм, задаваемых вероятностным образом и хорошо описывающих многие реальные формы, такие, как рельефы местности, береговые линии, сети рек, хлопья снега и ветви деревьев. Например, реалистичное изображение горы создается путем аппроксимации горы при помощи полигональной сетки. Каждый полигон, который необязательно является плоским, затем некоторое число раз рекурсивно подразделяется, чтобы создать неровный, с зазубринами, рельеф местности. Разбиение проводится с применением случайной функции. Таким образом, из начальной аппроксимации получается множество многоугольников. Далее проводится удаление скрытых поверхностей и применяется соответствующая модель закраски.

14.3. Цветовые модели.

14.3.1. Цветовая модель *RGB*

RGB (аббревиатура английских слов red, green, blue — красный, зелёный, синий) или КЗС — аддитивная **цветовая модель**, как правило, описывающая способ кодирования цвета для цветовоспроизведения. Выбор основных цветов обусловлен особенностями физиологии восприятия цвета сетчаткой человеческого глаза.

На принципе такого деления света основан цветной телевизор или монитор Вашего компьютера. Если говорить очень грубо, то монитор, в который Вы сейчас смотрите состоит из огромного количества точек (их количество по вертикали и горизонтали определяет разрешение монитора) и в каждую эту точку светят по три "лампочки": красная, зеленая и синяя. Каждая "лампочка" может светить с разной яркостью, а может не светить вовсе. Если светит только синяя "лампочка" - мы видим синюю точку. Если только красная - мы видим красную точку. Аналогично и с зеленой. Если все лампочки светят с полной яркостью в одну точку, то эта точка получается белой, так как все градации этого белого опять собираются вместе. Если ни одна лампочка не светит, то

точка кажется нам черной. Так как черный цвет - это отсутствие света. Сочетая цвета этих "лампочек", светящихся с различной яркостью можно получать различные цвета и оттенки.

Яркость каждой такой лампочки определяется интенсивностью (делением) от 0 (выключенная "лампочка") до 255 ("лампочка", светящая с полной "силой"). Такое деление цветов называется цветовой моделью RGB от первых букв слов "RED" "GREEN" "BLUE" (красный, зеленый, синий).

Таким образом **белый цвет** нашей точки в цветовой модели RGB можно записать в следующем виде:

R (от слова "red", красный) - 255

G (от слова "green", зеленый) - 255

B (от слова "blue", синий) - 255

"Насыщенный" красный будет выглядеть так:

R - 255

G - 0

B - 0

Чёрный:

R - 0

G - 0

B - 0

Желтый цвет будет иметь следующий вид:

R - 255

G - 255

B - 0

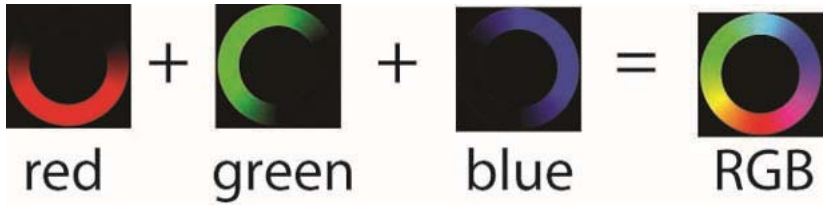
Так же, для записи цвета в rgb, используют шестнадцатеричную систему. Показали интенсивности запмсывают по порядку #RGB:

Белый - #ffffff

Красный - #ff0000

Черный - #00000

Желтый - #ffff00



Как мы отметили, в цветовом координатном пространстве *RGB* любой цвет получается как сумма (смешение) красного, зеленого и синего цветов. Если представить это пространство в виде куба, то на главной диагонали куба, образованного из нормированных компонентов, будут расположены серые цвета (ахроматические). Наряду с тем, что накоплен большой объем информации о реакции и чувствительности глаза к трем стимулам *RGB,* это цветовое пространство является аппаратно ориентированным. Цветные электронно - лучевые трубки и жидкокристаллические дисплеи отображают цветные изображения, основываясь на аддитивной смеси этих трех компонентов.

Аддитивной она называется потому, что цвета получаются путём добавления (англ. *addition*) к чёрному цвету. Иначе говоря, если цвет экрана, освещённого цветным прожектором, обозначается в **RGB** как

(r_1, g_1, b_1), а цвет того же экрана, освещенного другим прожектором, — (r_2, g_2, b_2), то при освещении двумя прожекторами цвет экрана будет обозначаться как (r_1+r_2, g_1+g_2, b_1+b_2).

Изображение в данной цветовой модели состоит из трёх каналов. При смешивании основных цветов (основными цветами считаются красный, зелёный и синий) — например, синего (B) и красного (R), мы получаем пурпурный (M magenta), при смешении зелёного (G) и красного (R) — жёлтый (Y yellow), при смешении зелёного (G) и синего (B) — циановый (C cyan). При смешении всех трёх цветовых компонентов мы получаем белый цвет (W white).

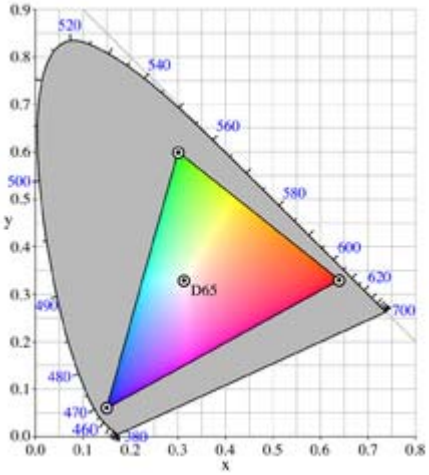
В телевизорах и мониторах применяются три электронных пушки (светодиода, светофильтра) для красного, зелёного и синего каналов.

Наиболее распространённое цветовое пространство sRGB, использующееся с цветовой моделью **RGB**, имеет по многим тонам цвета более широкий цветовой охват (может представить более насыщенные цвета), чем типичный охват цветов цветовых пространств в CMYK, поэтому иногда изображения, замечательно выглядящие в RGB, значительно тускнеют и гаснут в CMYK.

Определение

Цветовая модель RGB является зависимой от устройства. Поскольку мониторы разных моделей и производителей различаются, было предложено несколько стандартов цветовых пространств для этой модели. Например, sRGB является стандартом для изображения на мониторе (профиль «по умолчанию» для компьютерной графики). Также распространён Adobe RGB, а при редактировании используется ProPhoto.

Цветовая модель RGB может использовать разные базовые цвета (в том числе, цвета не реализуемые физически), разную цветовую температуру для «белой точки», и разный показатель гамма-коррекции.

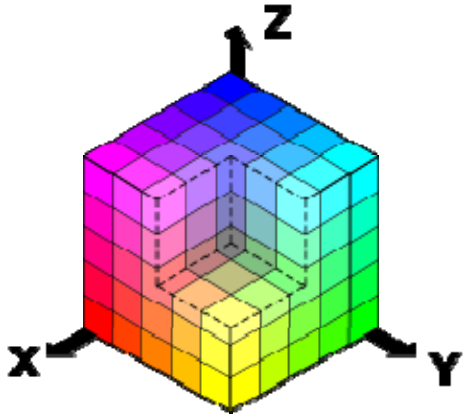


Ограничение sRGB по возможности передачи цветов

Числовое представление

RGB-цветовая модель, представленная в виде куба

Для большинства приложений значения координат r, g и b можно считать принадлежащими отрезку [0,1], что представляет пространство RGB в виде куба 1×1×1.



В компьютерах для представления каждой из координат представляются в виде одного октета, значения которого обозначаются для удобства целыми числами от 0 до 255 включительно, где 0 — минимальная, а 255 — максимальная интенсивность. В этом случае чаще используется гамма-компенсированое цветовое пространство sRGB, обычно с показателем 1,8 (Mac) или 2,2 (PC).

Вместе с тем, используются также 16 битный цвет (с диапазонами 0 — 65535 или 0 — 32768, в зависимости от конкретной реализации), а для изображений HDR — 32 битный цвет (в целых значения или в числах с плавающей запятой). В последнем случае возможны яркости «белее белого» и даже «отрицательные яркости», которые не выводятся на экран, но хранятся в памяти и учитываются при различной фильтрации.

В языке HTML используется запись вида «#rrggbb», называемая шестнадцатеричной: каждая координата записывается в виде трех шестнадцатеричных цифр («rr», «gg», «bb»), без пробелов (см. *цвета HTML*). Например, белый цвет кодируется строкой `#FFFFFF`.

COLORREF

COLORREF — стандартный тип для представления цветов в Win32. Используется для определения цвета в виде RGB. Размер — 4 байта. При определении какого-либо RGB цвета, значение переменной типа COLORREF можно представить в шестнадцатеричном виде так:

0x00bbggrr

rr, gg, bb — значение интенсивности соответственно красной, зелёной и синей составляющих цвета. Максимальное их значение — 0xFF.

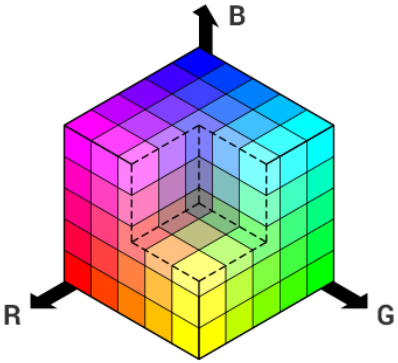
Определить переменную типа COLORREF можно следующим образом:

```
COLORREF C = RGB( r, g, b );
```

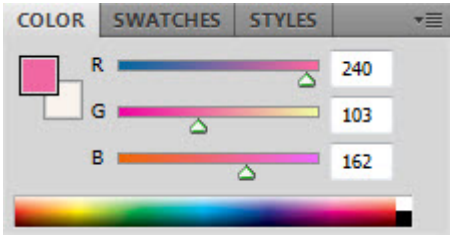
r, g и b — интенсивность (в диапазоне от 0 до 255) соответственно красной, зелёной и синей составляющих определяемого цвета C. То есть ярко-синий цвет может быть определён как (0,0,255), красный как

(255,0,0), ярко-фиолетовый — (255,0,255), чёрный — (0,0,0), а белый — (255,255,255)

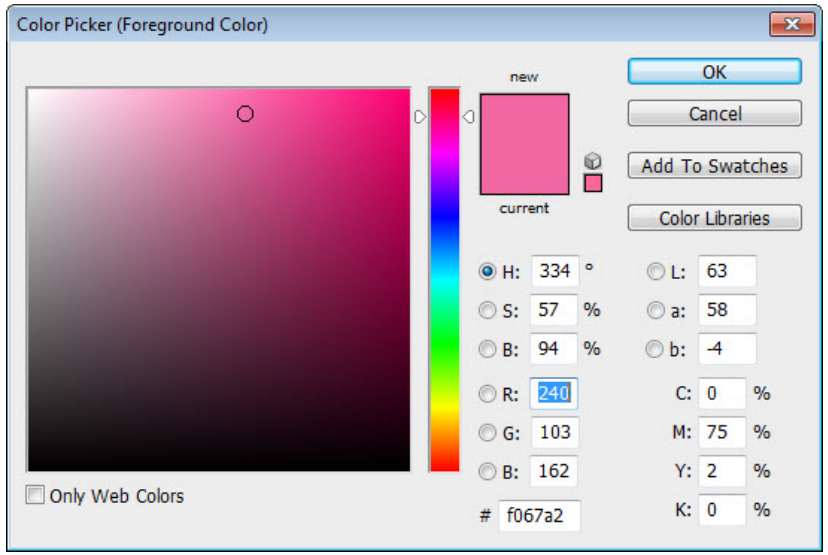
Поскольку в модели RGB есть три основные составляющие цвета, ее можно представить, как мы уже говорили, в виде куба. Получается, что каждая точка в пространстве этого куба (которую можно задать с помощью трех координат) — определенный цвет.



В компьютерах каждая из координат задается целым числом — от 0 до 255.



В HTML используется шестнадцатеричная запись: каждая координата задается двумя шестнадцатеричными числами. Вот, например, показанный выше цвет с RGB-координатами (240, 103, 162) в шестнадцатеричной записи выглядит так: #f067a2.



А вот как выглядит смешение цветов в числовом представлении:

красный	255	0	0
синий	0	0	255
пурпурный	255	0	255

синий	0	0	255
зеленый	0	255	0
голубой	0	255	255

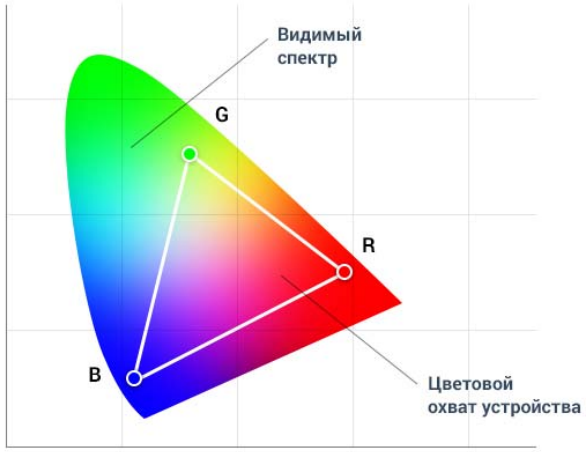
красный	255	0	0
зеленый	0	255	0
желтый	255	255	0

Ограничения модели RGB

На практике при применении модели RGB не всегда удается точно передать нужный цвет.

Первая проблема связана с технологией изготовления мониторов. Как уже упоминалось, цвет, воспроизводимый монитором, зависит от типа нанесенного на него люминофора. Но разными производителями используются разные типы люминофора. Кроме того, по мере старения монитора меняются качества люминофора и характеристик электронных прожекторов или светодиодов. Другими словами, на разных мониторах цвета могут немного различаться.

Вторая проблема имеет уже не технический характер, она проистекает из ограничений самого метода смешения цветов. Дело в том, что **с помощью аддитивного синтеза нельзя получить все цвета видимого спектра**. Все, что может монитор — это смешивать красный, зеленый и синий. Если обозначить эти цвета на диаграмме точками, то все множество цветов, которые можно получить их смешением, окажутся внутри получившегося треугольника. И площадь его, как мы видим, гораздо меньше, чем диапазон цветов, которые может различать человек.



14.3.2 Цветовая модель *YcrCb*

В этом пространстве компонент *Y* включает в себя только информацию о яркости пикселов, а компоненты *Cb* и *Cr* содержат только информацию о цвете и насыщенности. Поскольку органы зрения менее чувствительны к цвету предметов, чем к их яркости, такое

пространство позволяет передать компонент яркости с большим разрешением, чем компоненты цветности.

Определение компонентного сигнала задается через сигналы основных цветов *R, G, B* из уравнения, рекомендованного стандартом федеральной комиссии связи *(FCC)*:

$$Y=0{,}299\,R+0{,}587G+0{,}114B.$$
$$(R-Y)=R-0{,}299\,R-0{,}587G-0{,}114B=0{,}701R-0{,}587G-0{,}114B.$$
$$(B-Y)=B-0{,}299\,R-0{,}587G-0{,}114B=-0{,}299R-0{,}587G+0{,}886B.$$

(3.1)

Здесь *R, G, B* - исходные сигналы основных цветов, подвергнутых предварительной гамма - коррекции с целью обеспечения оптимального качества изображения на экране кинескопа. Если значения сигналов привести к единице (максимальный уровень сигнала - 1В), то получим значения для белого, черного и насыщенных основных и дополнительных цветов, представленные в таблице 3.1.

Таблица 3.1

Цвет	*R*	*G*	*B*	*Y*	*Cr=R-Y*	*Cb= B-Y*
Белый	1,0	1,0	1,0	1,0	0	0
Черный	0	0	0	0	0	0
Красный	1,0	0	0	0,299	0,701	-0,299
Зеленый	0	1,0	0	0,587	-0,587	-0,587
Синий	0	0	1,0	0,114	-0,114	0,886
Желтый	1,0	1,0	0	0,886	0,114	-0,886
Голубой	0	1,0	1,0	0,701	-0,701	0,299
Пурпурный	1,0	0	1,0	0,413	0,587	0,587

Значения сигнала *Y* находятся в пределах от 0 до 1, значения цветоразностных сигналов изменяются от − 0,701 до 0,701 для *Cr* и от -0,886 до 0,886 для *Cb.* Приведение диапазонов изменения цветоразностных сигналов к единице достигается введением нормирующих коэффициентов

$$Kr=0{,}5/0{,}701=0{,}713,\ Kb=0{,}5/0{,}866=0{,}564.$$

Сигнал яркости и нормированные цветоразностные сигналы связаны с сигналами основных цветов следующим матричным преобразованием:

$$\begin{bmatrix} Y \\ Cr \\ Cb \end{bmatrix} = \begin{bmatrix} 0{,}299 & 0{,}587 & 0{,}114 \\ 0{,}5 & -0{,}41869 & 0{,}08131 \\ -0{,}168874 & -0{,}33126 & 0{,}5 \end{bmatrix} \times \begin{bmatrix} R \\ G \\ B \end{bmatrix}. \quad \textbf{(3.2)}$$

Переход от цветового координатного пространства *RGB* к пространству *YCrCb* (3.1) соответствует рекомендациям Международного телекоммуникационного союза *(ITU- International Telecommunication Union) ITU- Tc* идентификатором *BT.601.*

Определение компонентного сигнала задается через сигналы основных цветов *R, G, B.* При 8-ми разрядном представлении компонентов диапазон значений составляет для *Y* [0,255], а для *Cr* и *Cb*− [−128,127]. После перевода цветоразностных сигналов в диапазон [−128,127], получим матрицу для основных цветов, определяющую обратное преобразование в соответствии с уравнениями:

$$\begin{bmatrix} R \\ G \\ B \end{bmatrix} = \begin{bmatrix} 1 & 1{,}402 & 0 \\ 1 & -0{,}714 & -0{,}344 \\ 1 & 0 & 1{,}772 \end{bmatrix} \times \begin{bmatrix} Y \\ Cr \\ Cb \end{bmatrix}. \quad \textbf{(3.3)}$$

Это пространство используется в ТВ системах *PAL* и *SECAM,* а также при кодировании неподвижных изображений и видеопоследовательностей. При формировании сигнала используется сокращение избыточности цветоразностных сигналов. Этот принцип основан на особенности человеческого зрения не различать или плохо различать цвета мелких деталей изображения. Экспериментально было установлено [20], что при расстоянии до экрана цветного телевизора *L=4,5h,* где *h* - высота экрана, наблюдатель не ощущает мелкие синие детали как цветные при пространственной частоте этих деталей > 0,5 − 0,6 МГц, а красные − при частоте > 1,3 − 1,5 МГц.

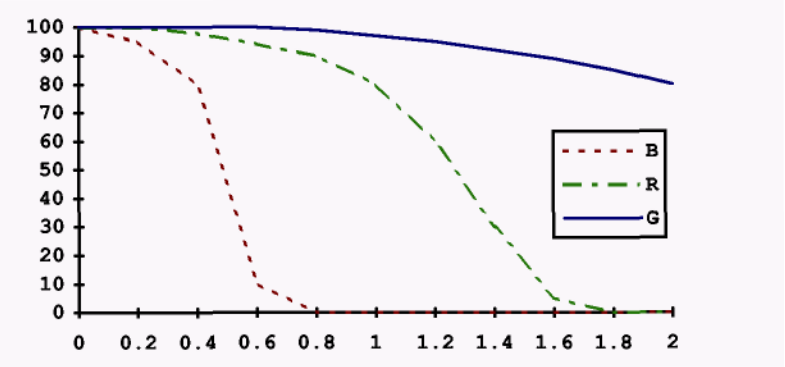


Рис. 3.3 Графики зависимости видимой насыщенности от размеров деталей и их цветов.

На этой особенности человеческого зрения основано построение аналоговых и цифровых систем цветного телевидения, в которых частота дискретизации сигнала яркости в 2 раза превышает частоту дискретизации каждого из цветоразностных каналов. В форматах 4:2:2 и 4:2:0 вводится по одному цветоразностному отсчету на 2 отсчета яркости.

Известно, что органы зрения человека менее чувствительны к цвету предметов, чем к их яркости. В цветовом пространстве RGB все три компонента считаются одинаково важными, и они обычно сохраняются с одинаковым разрешением. Однако можно отобразить цветовое изображение более эффективно, отделив светимость от цветовой информации и представив ее с большим разрешением, чем цвет. Поэтому цветовое пространство YCbCr и его вариации является популярным методом эффективного представления цветных изображений.

Буква Y в таких цветовых пространствах обозначает компоненту светимость, которая вычисляется как взвешенное усреднение компонент R, G и B по следующей формуле:

$$Y = k_r R + k_g G + k_b B$$

где k_r обозначает соответствующий весовой множитель. Остальные цветовые компоненты по существу определяются в виде разностей между светимостью Y и компонентами R, G и B:

$$Cb = B - Y,$$

$$Cr = R - Y,$$

$$Cg = G - Y.$$

При этом получаются четыре компоненты нового пространства вместо трех RGB. Однако число Cb+Cr+Cg является постоянным, поэтому только две из трех хроматических компонент необходимо хранить, а третью вычислять на основе них. Чаще всего в качестве две искомых цветовых компонент используют Cb и Cr. Преимущество пространства YCbCr по сравнению с RGB заключается в том, что Cb и Cr можно представлять с меньшим разрешением, чем Y, т.к. глаз человека менее чувствителен к цвету предметов, чем к их яркости. Это позволяет сократить объем информации, требуемый для представления хроматических компонент, без заметного ухудшения качества передачи цветовых оттенков изображения. Такой подход к преобразованию цветового пространства дает дополнительный эффект при сжатии цветных изображений. При этом алгоритмы сжатия сначала преобразуют исходное цветовое пространство из RGB в YCbCr, сжимают, а затем при восстановлении обратно преобразуют изображение в цветовое пространство RGB, т.к. оно используется в ЭВМ. При этом формулы для прямого и обратного преобразований выглядят следующим образом:

$$\left.\begin{aligned} Y &= k_r R + (1 - k_b - k_r) G + k_b B, \\ Cb &= \frac{0,5}{1 - k_b}(B - Y), \\ Cr &= \frac{0,5}{1 - k_r}(R - Y). \end{aligned}\right\}$$

прямое преобразование

$$\left.\begin{aligned} R &= Y + \frac{1-k_r}{0,5} Cr, \\ G &= Y - \frac{2k_b(1-k_b)}{1-k_b-k_r} C_b - \frac{2k_r(1-k_r)}{1-k_b-k_r} C_r, \\ B &= Y + \frac{1-k_b}{0,5} C_b. \end{aligned}\right\}$$ обратное преобразование

Отметим, что множитель k_g получается из соотношения $k_g + k_r + k_b - 1$, а величина компоненты G получается вычитанием суммы Cb и Cr из Y.

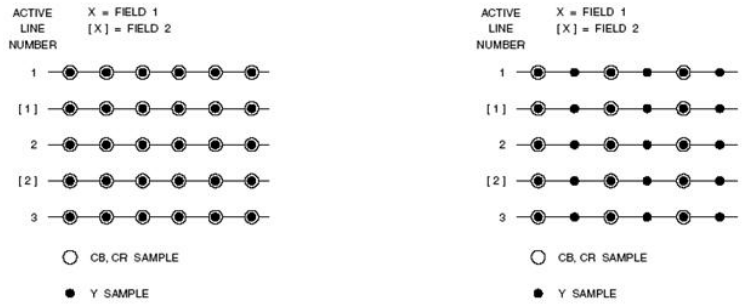
Рекомендация ITU-T предлагает следующие коэффициенты: $k_b = 0,114$ и $k_r = 0,229$. Используя эти значения в данных уравнениях, получаем широко распространенные формулы:

$$\begin{aligned} Y &= 0,299R + 0,587G + 0,114B, \\ Cb &= 0,564(B - Y), \\ Cr &= 0,713(R - Y); \end{aligned}$$

$$\begin{aligned} R &= Y + 1,402Cr, \\ G &= Y - 0,344Cb - 0,714Cr, \\ B &= Y + 1,772Cb. \end{aligned}$$

Как отмечалось выше хроматические компоненты Cb и Cr могут быть представлены с меньшим разрешением, чем световая компонента Y. При этом на практике используют следующие форматы их взаимного представления.

Самый очевидный формат это так называемый формат 4:4:4, который означает полную точность в передаче хроматических компонент, т.е. на каждые 4 световые отсчеты Y передаются по 4 отсчета компонент Cb и Cr (рис. 3.4 а).

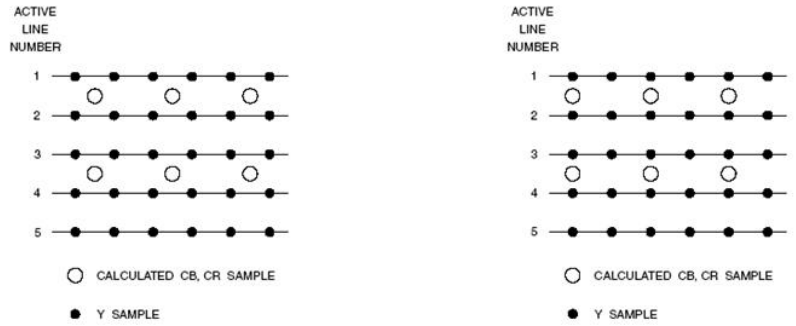


а) б)

Рис. 3.4. Расположение хроматических компонент

Другой формат 4:2:2 (YUY2) предполагает, что на каждые 4 отсчета компоненты Y приходится по два отсчета хроматических компонент, расположение которых представлено на рис. 3.4, б. Данный формат используется для высококачественного цветного видео и используется в стандартах MPEG-4 и H.264.

Наиболее популярный формат сэмплирования 4:2:0 (YV12) каждая компонента Cb и Cr имеет один отсчет на 4 отсчета Y (рис. 3.5 а, б). Причем отсчеты компонент Cb и Cr, как правило, вычисляются двумя способами. В первом случае выполняется интерполяция по 4 ближайшим отсчетам компонент Cb и Cr для формирования одного отсчета для них (рис. 3.5, а). Такой подход применяется в стандартах MPEG-1 и H.261, H.263. В другом случае выполняется интерполяция по двум вертикальным отсчетам (рис. 3.5, б) и применяется в стандарте MPEG-2.



а) б)

Рис. 3.5. Представление формата 4:2:0

Благодаря экономичному представлению цветных сцен, формат 4:2:0 широко используется во многих потребительских приложениях, таких как видеоконференции, цифровое телевидение, DVD. Поскольку хроматические компоненты отбираются в 4 раза реже компонент яркости, то пространство 4:2:0 YCbCr занимает в 2 раза меньше отсчетов по сравнению с форматом видео 4:4:4 RGB.

14.3.3. Цветовая модель CMYK

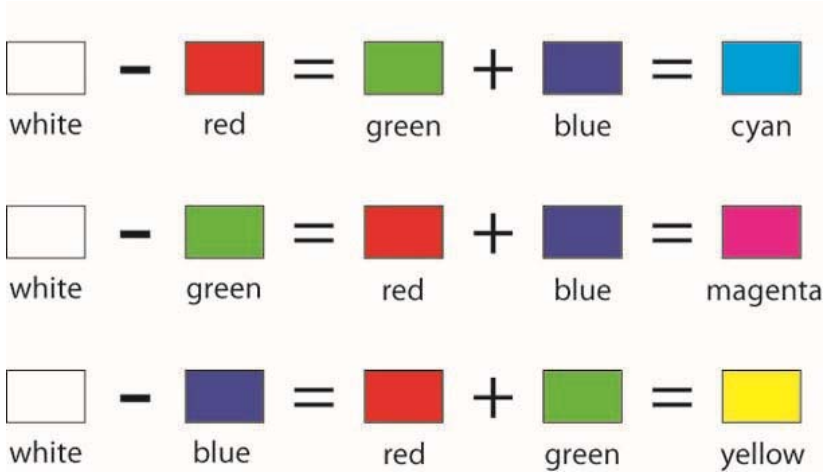
Итак, теперь мы знаем, каким способом наш компьютер передает нам цвет той или иной точки. Давайте теперь воспользуемся приобретенными знаниями и попробуем получить белый цвет с помощью красок. Для этого купим в магазине гуашь, возьмем баночки с красной, синей и зеленой краской, и смешаем их. Получилось? Нет.

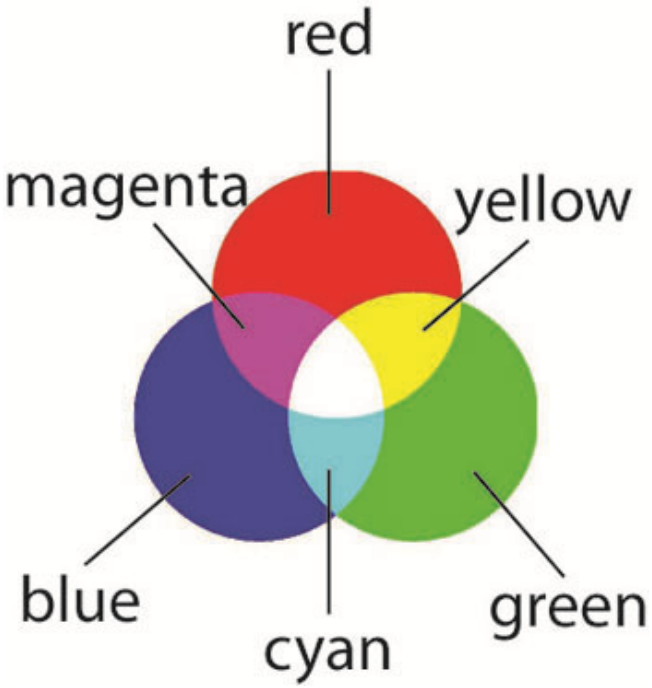
Проблема в том, что наш монитор излучает свет, то есть светится, но в природе многие объекты не обладают таким свойством. Они попросту отражают белый свет, который на них падает. Причем если предмет отражает весь спектр белого света, то мы видим его белым, а если же часть этого света им поглощается - то не совсем.

Примерно так: мы светим на красный предмет белым светом. Белый свет можно представить как R-255 G-255 B-255. Но предмет не хочет отражать весь свет, который мы на него направили, и "ворует" все

оттенки зеленого и синего. В итоге отражает только R-255 G-0 B-0. Именно поэтому он нам и кажется красным.

Так что для печати на бумаге весьма проблематично пользоваться цветовой моделью RGB. Для этого, как правило, используется цветовую модель CMY (цми) или CMYK (цмик). Цветовая модель CMY основана на том, что сам по себе лист бумаги белый, то есть отражает практически весь спектр RGB, а краски, наносимые на нее, выступают в качестве фильтров, каждый из которых "ворует" свой цвет (либо red, либо green, либо blue). Таким образом цвета этих красок определяются вычитанием из белого по одному цветов RGB. Получаются цвета Cyan (что-то вроде голубого), Magenta (можно сказать, розовый), Yellow (желтый).





Если в цветовой модели RGB градация каждого цвета происходила по яркости от 0 до 255, то в цветовой модели CMYK у каждого цвета основным значением является "непрозрачность" (количество краски) и определяется процентами от 0% до 100%.

Таким образом, белый цвет можно описать так:

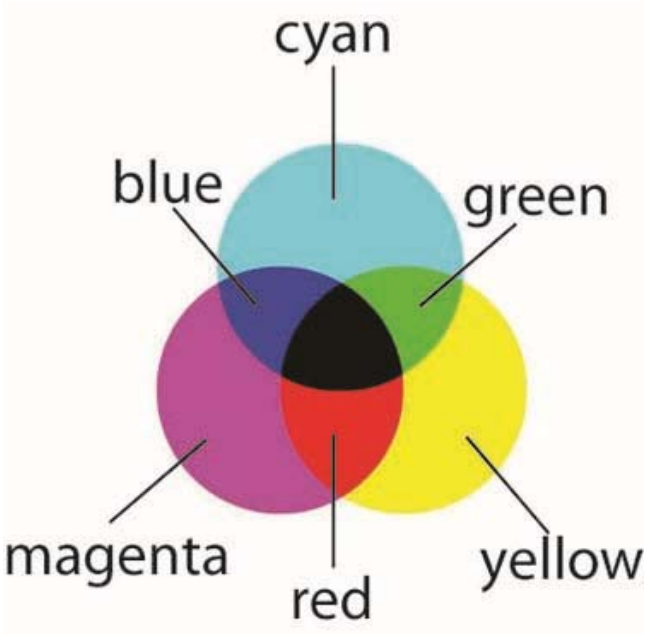
C (cyan) - 0%; M (magenta) - 0%; Y (yellow) - 0%.

Красный - C-0%; M-100%; Y-100%.

Зеленый - C-100%; M-0%; Y-100%.

Синий - C-100%; M-100%; Y-0%.

Черный - C-100%; M-100%; Y-100%.



Однако, это возможно только в теории. А на практике же обойтись цветами CMY не получается. И черный цвет при печати получается скорее грязно-коричневым, серый не похож сам на себя, а темные оттенки цветов создать проблематично. Для урегулирования конечного цвета используется еще одна краска. Отсюда и последняя буква в названии CMYK (ЦМИК). Расшифровка этой буквы может быть разной: это может быть сокращение от blacK (черный). И в сокращении используется именно последняя буква, чтобы не спутать этот цвет с цветом Blue в модели RGB. Печатники очень часто употребляют слово "Контур" относительно этого цвета. Так что возможно, что буква K в абревиатуре CMYK (ЦМИК) - это сокращение от немецкого слова "Kontur". Так же это может быть сокращение от Key-color (ключевой цвет). Однако ключевым его назвать сложно, так как он является скорее дополнительным. И на черный этот цвет не совсем похож. Если печатать только этой краской изображение получается скорее серое.

Поэтому некоторые придерживаются мнение, что буква K в обревиатуре CMYK означает "Kobalt" (темно-серый, нем.).

Как правило, используется для обозначения этого цвета термин "black" или "черный".

Печать с использованием цветов CMYK называют "полноцветной" или "триадной".

Стоит, наверное, сказать, что при печати CMYK (ЦМИК) краски не смешиваются. Они ложатся на бумагу "пятнами" (растром) одна рядом с другой и смешиваются уже в воображении человека, потому что эти "пятна" очень малы. То есть изображение растрируется, так как иначе краска, попадая одна на другую, расплывается и образуется муар или грязь. Существует несколько разных способов растрирования.

14.3.4. Цветовая модель grayscale

Изображение в цветовой модели grayscale многие ошибочно называют черно-белым. Но это не так. Черно-белое изображение состоит только из черных и белых тонов. В то время, как grayscale (оттенки серого) имеет 101 оттенок. Это градация цвета Kobalt от 0% до 100%.

Черно-белое изображение

Изображение grey-scale

Модель **Grayscale (Серая шкала)** применяется для отображения черно-белых фотографий или подобных фотографиям изображений в черно-белой полиграфии. Традиционная серая шкала, использующая для хранения информации о каждом пикселе изображения один байт,

может передавать 256 оттенков (градаций) серого цвета или яркости (brightness): значение 0 представляет черный цвет, а значение 255 — белый. Шкала Grayscale выражается в процентах, в этом случае 0% означает белый цвет (отсутствие краски на белой бумаге), а 100% — черный цвет.

Для черно-белой печати (газет, книг, черно-белого принтера) рисунок желательно перевести в Grayscale. И очень внимательно на него посмотреть. То, что красиво смотрится в цвете, после перевода в ч/б может вдруг стать совершенно непригодным - мутным и неразборчивым. Причины этого понять нетрудно: *когда соседние участки изображения имеют разные цвета, мы отлично их различаем, даже если яркости этих участков очень близки. А когда изображение становится черно-белым, участки с одинаковой яркостью просто не различимы.*

Перевод цветного изображения в черно-белое

Достаточно часто великолепная цветная фотография, будучи переведенной в черно-белый режим, становится совершенно не контрастной. Можно проделать эксперимент: создать файл, состоящий из шести цветных квадратов:

И перевести его в черно-белый режим (Image/ Mode/ Grayscale - Рисунок/ Режим/ Градации серого):

После конвертации черно-белое изображение должно быть четким и разборчивым, к сожалению с двумя квадратами размещенными слева и двумя по центру это не удалось – они слились.

Дело в том, что в цветном варианте между красным и синим мы видим резкий цветовой контраст, которого в черно-белом изображении быть не может. В центре мы видим большой контраст по цветовой насыщенности между верхним и нижним квадратами. Но информация о цвете в черно-белом изображении теряется и результат преобразования в этом случае тоже неудачен. В третьем столбце контраст относится не к цвету и насыщенности а к яркости, а вот яркость для черно-белого изображения – самый важный параметр, это единственный тип контраста который может быть учтен.

Итак, **если цветное изображение изобилует яркостными контрастами, то мы успешно переведем его в черно-белое, а вот если нет, то мы должны предпринять дополнительные действия.**

Редактор Photoshop при переводе изображения в черно-белое каждому пикселю передает 30% красного (R), 60% зеленого (G), и 10% синего (B) каналов от цветного варианта. Это называется усредненный метод с весами, даже если изображение не в RGB, Photoshop все равно переводит его в RGB, а уже затем обрабатывает. Вариантов с помощью чего можно повысить контраст в Photoshop, великое множество, это могут быть Levels (уровни), Channel Mixer (смешивание каналов), различные режимы наложения слоев.

Чтобы сделать из цветной фотографии чёрно-белую Photoshop предлагает несколько способов. Какой вам подойдёт в наибольшей степени - зависит от того, какие цели вы преследуете.

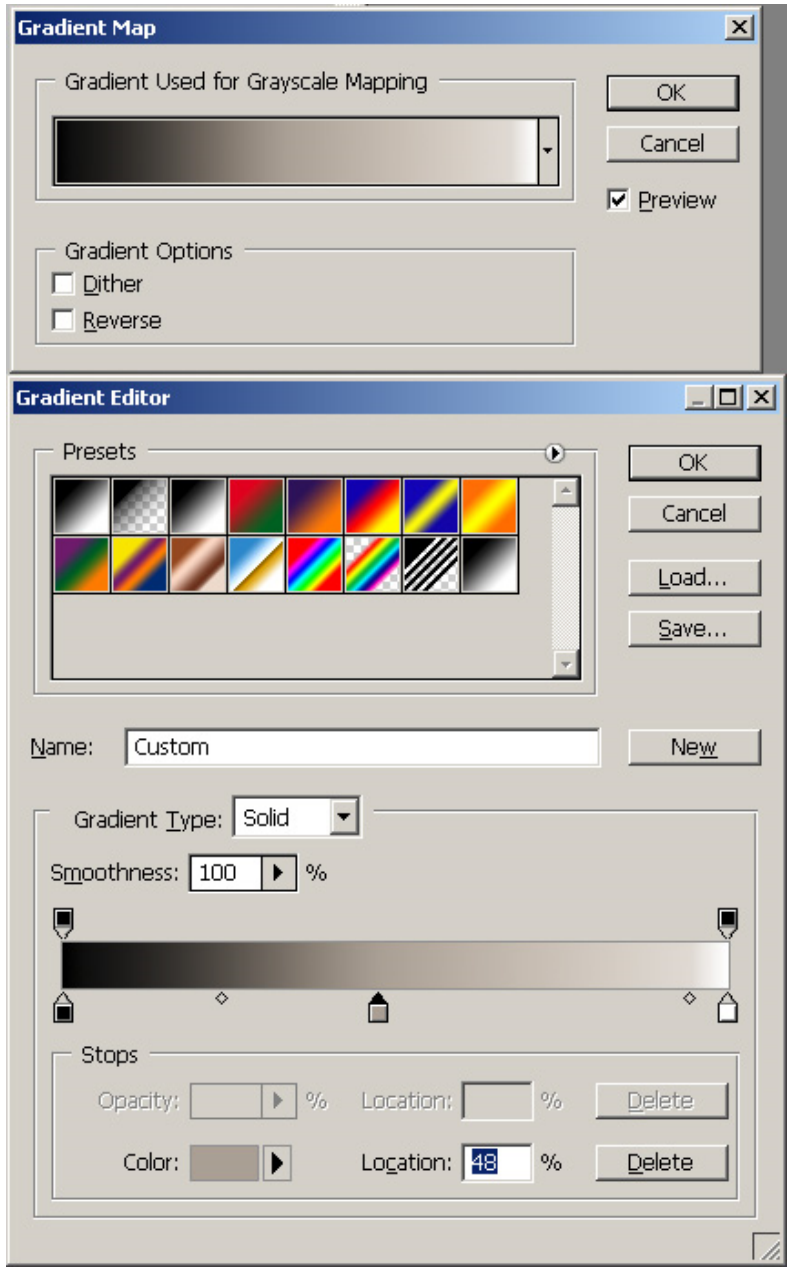
Grayscale и Desaturate

Самый простой способ это применить команду **Image-Mode-Grayscale (Изображение-Режим-Серая шкала)**. Обратите внимание, что этот метод переведет все слои текущего изображения в чёрно-белый режим. Плюсы этого метода заключаются в том, что он быстр, не образует шумов, сохраняет общую яркость и контраст изображения. К минусам можно отнести отсутствие контроля оттенков, потеря данных о цвете, невозможность создания корректирующего слоя.

Если же вам нужно перевести в ч/б только один слой, то аналогом применения команды **Grayscale** будет команда **Image-Adjustments-Desaturate (Изображение-Коррекция-Обесцветить)**. Этот способ достаточно грубый и редко когда, используя его, удаётся добиться хороших результатов. Но его можно использовать как грубую и быструю оценку, стоит ли переводить изображение в ч/б или нет. Минусы Desaturate: нарушение яркости по сравнению с исходником, появление яркостных пятен, шумов, отсутствие контроля оттенков, потеря данных о цвете, невозможность создания в виде корректирующего слоя.

Gradient Map

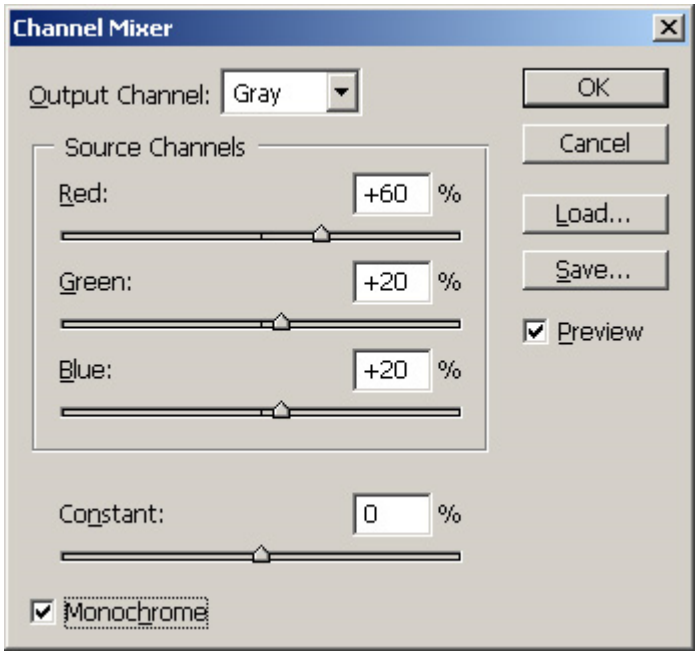
Еще один способ - воспользоваться командой **Image-Adjustments-Gradient Map (Изображение/Коррекция/Карта градиента)**. Этот способ дает уже результат получше и его можно в некоторой степени контролировать. К его плюсам отнесём чистоту, сохранение данных о цвете, возможность создания в виде корректирующего слоя, а к минусам - повышение контраста по сравнению с оригиналом (хотя, для кого-то это может быть и плюсом).



Если кликнуть по градиенту в окне **Gradient Map**, то можно изменить градиент: сдвинув черный маркер, можно увеличить или уменьшить долю черного цвета, сдвинув белый - белого, можно добавить дополнительный цвет и тогда фотография станет не только черно-белой, но и тонированной. А если выбрать другие цвета, то можно получить совсем психоделические картины.

Channel Mixer

Одним из самых мощных инструментов Photoshop'а для перевода цветного изображения в ч/б является команда **Image-Adjustments-Channel Mixer (Изображение- Коррекция-Микширование каналов).**



Для начала в окне **Channel Mixer** необходимо поставить галочку напротив опции **Monochrome**. После этого можно преступать к работе с бегунками красного, зеленого и синего каналов. Здесь вас теперь ждет полная свобода творчества, изменяя положения бегунков можно добиться любых эффектов, например, повышая значения красного канала мы добьемся затемнения синих участков изображения, что, например, можно использовать при переводе в ч/б фотографий с небом. Как показывает практика хороших результатов можно добиться, если сумма значений всех бегунков будет равна или около 100. Но в некоторых ситуациях и это правило можно нарушить - главное, какого эффекта вы хотите достичь.

Этот метод предоставляет хороший контроль над яркостью и контрастом, над оттенками разных элементов изображения и его можно использовать в виде корректирующего слоя.

Исходное изображение и результаты применения Channel Mixer с параметрами 80/20/0 и -20/-30/150

Канал Lightness в цветовой схеме Lab

Если перевести изображение в Lab-пространство **Image - Mode - Lab Color (Изображение - Режим - Lab)** и переключиться на панель Channels, то мы увидим, что каналы вместо привычных Red, Green и Blue стали Lightness, a и b. Канал Lightness представляет из себя монохромное изображение, поэтому можно оставить видимым только его и перевести изображение в градации серого известной нам уже командой (Image - Mode - Grayscale). Этот метод позволяет переводить изображение в ч/б без образования шума, но повышает яркость и контраст, так как не учитывает информацию о цвете. Кроме этого к минусам отнесём невозможность создания в виде корректирующего слоя.

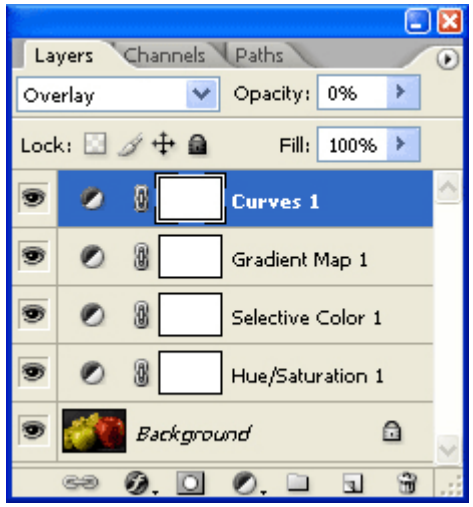
Чтобы применить этот метод, изображение нужно перевести в цветовое пространство Lab, после чего выбрать канал L в списке каналов или через Ctrl+3. Затем перевести его в Grayscale и утвердительно ответив на вопрос, игнорировать ли остальные каналы.

Black/White

Этот способ появился в версии Photoshop CS3. С помощью диалогового окна **Black/White (Черное/Белое)** можно определять яркость цветовых тонов. Результаты работы Black/White можно увидеть, немного поэкспериментировав с каретками цветовых оттенков. При значении 50 для определенных цветовых тонов все пиксели цвета принимают такие же значения яркости, как при использовании Hue/Saturation. При значении 0 соответствующие насыщенные цвета становятся черными, при значении 100 — белыми, а малонасыщенные цвета, соответственно, затемняются или становятся светлее. При отрицательных значениях затемняются также малонасыщенные и светлые пиксели этого цветового тона, при значениях выше 100 ненасыщенные и темные участки становятся светлее. Несмотря на то что Black/White напоминает Channel Mixer, принцип его действия совершенно другой: любые изменения в окне Channel Mixer влияют на все изображение, в то время как Black/White — только на определенные цветовые оттенки.

Комбинирование различных методов

На практике часто используются способы перевода в ч/б, которые комбинируют в себе перечисленные. Комбинированый способ - наверное, самый качественный способ получения ч/б изображения. Он заключатся в создании 4-х корректирующих слоёв. Создадим корректирующие слои как на картинке **(Слой/ Новый корректирующий слой)**, в той же последовательности, все в нормальном режиме смешивания, кроме **Curves** - этому слою задаем тип наложения **Soft Light** или **Overlay**.



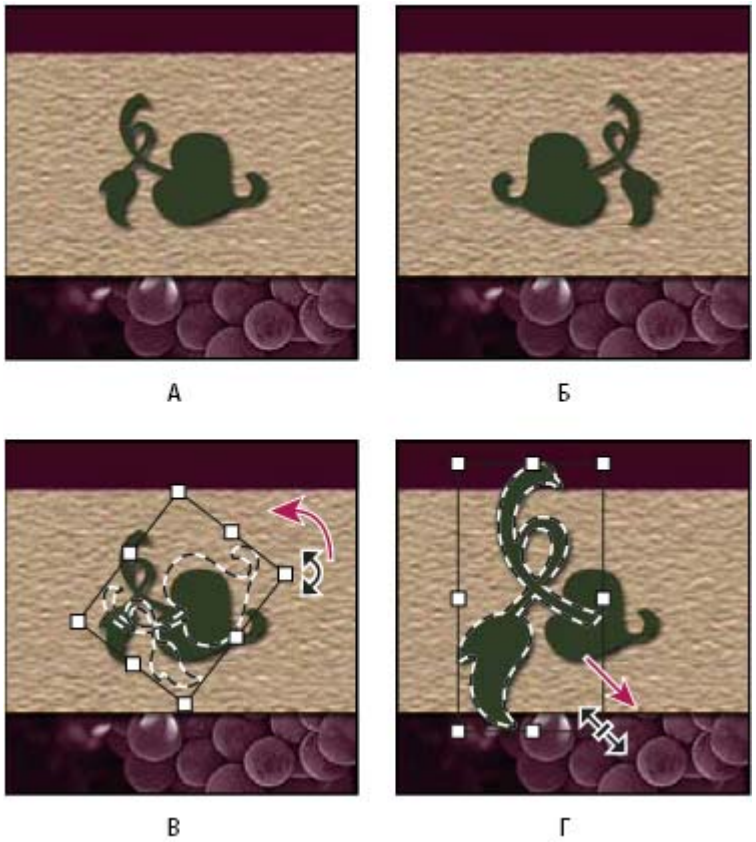
В слое **Hue/Saturation**, двигая ползунок Hue, меняем тон цветов, тем самым меняются итоговые оттенки яркости. Можно двигать ползунок Saturation, желательно в диапазоне значений -25+25, в больших значениях велик риск появления шумов и пятен. В слое **Selective Color** из выпадающего меню можем выбрать любой основной цвет и добавлять к нему другие, опять же влияя на итоговое изображение, каждого цвета в отдельности. Не забывайте, что ослабить эффект этих слоёв можно, уменьшив их непрозрачность. Слой **Gradient Map** (Вместо Gradient Map может быть использован **Channel Mixer)** - переводит наше изображение в ч.б. Слоем Curves можем подрегулировать итоговый контраст изображения установив нужное нам значение непрозрачности.

Минус способа один - не очень быстрая работа, хотя за качество всегда платили временем.

В большинстве случаев для работы с фотографиями самым удобным будет использование диалога Black/White или простой перевод в Grayscale. Если после этого добавить контраст с помощью кривых Curves (Ctrl+M), то мы получим достаточно хорошие ч/б изображения при минимальной затрате времени и сил.

Применение трансформирования

Трансформирование можно применять к выделенной области, целому слою, нескольким слоям. При работе с пикселами трансформирование влияет на качество изображения. Чтобы выполнить трансформирование, выделите нужный объект, затем выберите команду трансформирования. (Редактирование/ Трансформирование)



Трансформирование изображения

А. Исходное изображение

Б.Отраженный слой

В. Повернутая граница выделенной области

Г. Часть объекта в измененном масштабе

Команды подпунктов меню трансформирования

Масштабирование - Увеличение или уменьшение элемента относительно контрольной точки – заданной точки, вокруг которой выполняется трансформирование. Масштабировать можно по горизонтали или по вертикали, а также по горизонтали и по вертикали одновременно.

Поворот - Поворот элемента вокруг контрольной точки. По умолчанию эта точка находится в центре объекта. Однако ее можно переместить.

Наклон - Наклон элемента по вертикали или по горизонтали.

Искажение - Растягивание элемента по всем направлениям.

Перспектива - К выбранному элементу применяется перспектива схождения в одной точке.

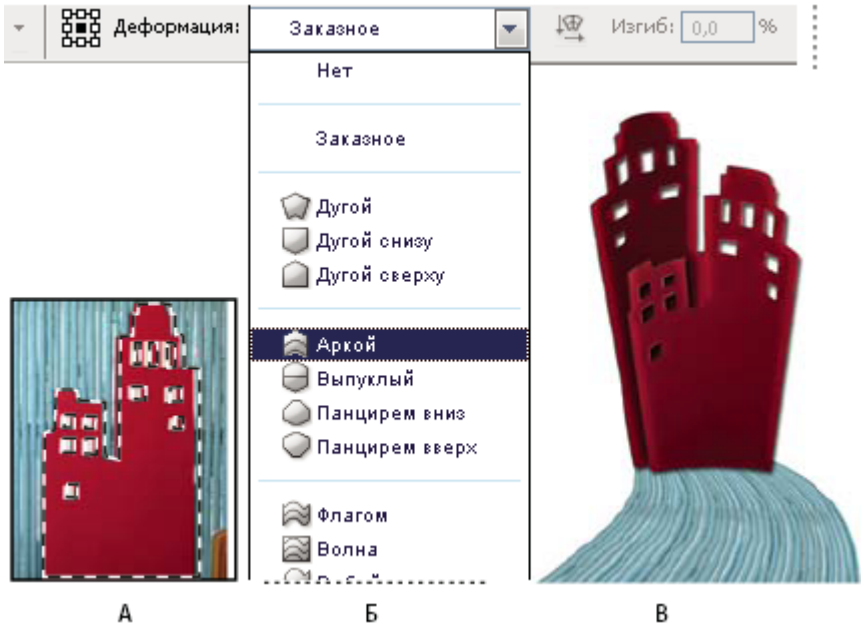
Деформация - Изменение формы элемента.

Отразить - Отражение выбранной области в горизонтальной или вертикальной плоскости.

Деформация элемента

Команда "Деформация" позволяет перетаскиванием опорных точек изменить форму изображения, фигур, контуров и т.д. Кроме того, деформацию можно выполнять с помощью нужной формы в выпадающем меню "Стиль деформации" на панели выбора параметров. Формы в выпадающем меню "Стиль деформации" можно изменять перемещением их опорных точек.

При искажении элемента с помощью опорных точек сетку деформации или опорные точки можно отображать и скрывать с помощью команды "Вспомогательные элементы" в меню "Редактирование".



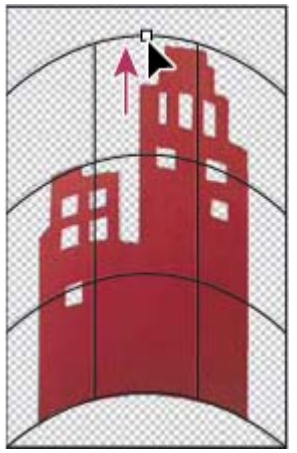
Использование деформации

А. Выбор фигуры для деформации

Б. Выбор формы в выпадающем меню "Стиль деформации" на панели выбора параметров.

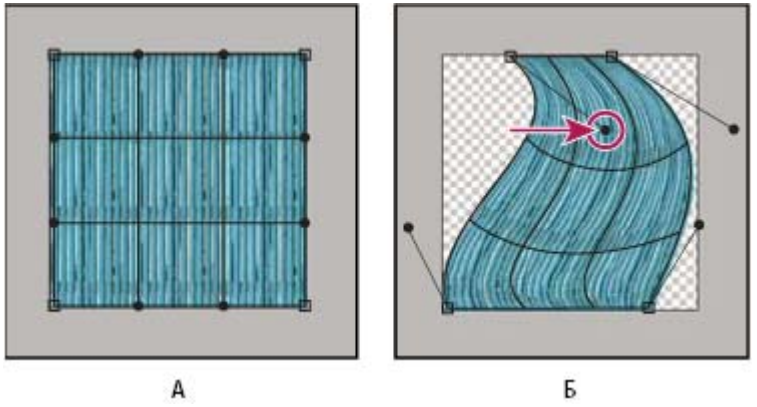
В. Результат использования нескольких параметров деформации

Для применения деформации выберите объект для деформации. В меню **"Редактирование" выберите "Трансформирование" > "Деформация".** Для деформации объекта с помощью определенной формы выберите эту форму в всплывающем меню "Стиль деформации" на панели выбора параметров.



Перетаскивание опорных точек для деформации сетки

Для манипуляций с фигурой можно перетаскивать опорные точки, сегмент ограничительной рамки или сетки, а также область внутри сетки. Кривая настраивается с помощью рукоятей опорных точек. Это аналогично настройке рукоятей искривленного сегмента в векторной графике. Чтобы отменить последнюю коррекцию рукояти, выберите в меню "Редактирование" пункт "Отменить".



Работа с формой деформации

А. Исходная сетка деформации

Б. Настройка рукоятей, сегментов сетки и областей внутри сетки

Нажмите клавишу "Ввод" для применения деформации.

Заметим, что каждый раз при трансформировании резкость растрового изображения (в отличие от формы или контура) немного снижается, поэтому рекомендуется выполнить несколько команд и только потом подтверждать суммарное преобразование, а не подтверждать все преобразования по отдельности.

Аппаратно-зависимые и аппаратно-независимые цветовые модели

Цветовые модели CMYK и RGB являются аппаратно-зависимыми, то есть они зависят от способа передачи нам цвета. Они указывают конкретному устройству, как использовать соответствующие им красители, но не имеют сведений о восприятии конечного цвета человеком. В зависимости от настроек яркости, контрастности и резкости монитора компьютера, освещенности помещения, угла, под которым мы смотрим на монитор, цвет с одними и теми же параметрами RGB воспринимается нами по-разному. А восприятие человеком цвета в цветовой модели "CMYK" зависит от еще большего ряда условий, таких как свойства запечатываемого материала (например, глянцевая бумага впитывает меньше краски, чем матовая, соответственно цвета на ней получаются более яркие и насыщенные), особенности краски, влажности воздуха, при котором сохла бумага, характеристик печатного станка…

Чтобы передать человеку более достоверную информацию о цвете, к аппаратно-зависимым цветовым моделям прикрепляют так называемые цветовые профили. Каждый из такого профиля содержит информацию о конкретном способе передачи человеку цвета и регулирует конечный цвет с помощью добавления или изъятия из какого-либо составляющего первоначального цвета параметров. Например, для печати на глянцевой пленке используется цветовой профиль, убирающий 10% Cyan и добавляющий 5% Yellow к

первоначальному цвету, из-за особенностей конкретной печатной машины, самой пленки и прочих условий. Однако даже прикрепленные профили не решают всех проблем передачи нам цвета.

Аппаратно-независимые цветовые модели не несут в себе сведений для передачи цвета человеку. Они математически описывают цвет, воспринимаемый человеком с нормальным цветным зрением.

14.3.5. Цветовые модели HSB и HLS

В основе этого цветового пространства лежит уже знакомое нам радужное кольцо RGB. Цвет управляется изменением таких параметров, как:

Hue - оттенок или тон;

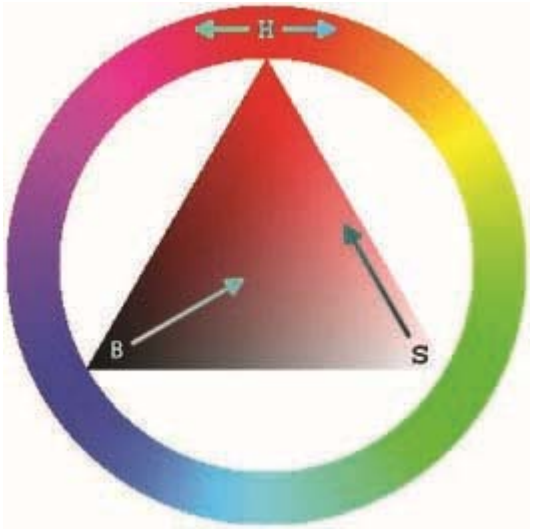
Saturation - насыщенность цвета;

Brightness - яркость.

Параметр hue - это цвет. Определяется градусами от 0 до 360 исходя из цветов радужного кольца.

Параметр saturation - процент добавления к этому цвету белой краски имеет значение от 0% до 100%.

Параметр Brightness - процент добавления черной краски так же изменяется от 0% до 100%.



Принцип похож на одно из представлений света с точки зрения изобразительного искусства. Когда в уже имеющиеся цвета добавляют белую или черную краску.

Это самая простая для понимания цветовая модель, поэтому ее очень любят многие web-дизайнеры. Однако она имеет ряд недостатков:

Глаз человека воспринимает цвета радужного кольца, как цвета, имеющие различную яркость. Например, спектральный зелёный имеет большую яркость, чем спектральный синий. В цветовой модели HSB все цвета этого круга считаются обладающими яркостью в 100%, что, к сожалению, не соответствует действительности.

Так как в её основе лежит цветовая модель RGB, она, все же является аппаратно-зависимой.

Эта цветовая модель конвертируется для печати в CMYK и конвертируется в RGB для отображения на мониторе. Так что догадаться, каким у вас в конечном счете получится цвет бывает весьма проблематично.

Аналогична этой модели цветовая модель HLS (расшифровка: hue, lightness, saturation).

Иногда используются для коррекции света и цвета в изображении.

14.3.6. Цветовая модель LAB

В этой цветовой модели цвет состоит из:

Luminance - освещенность. Это совокупность понятий яркость (lightness) и интенсивность (chrome)

A - это цветовая гамма от зеленного до пурпурного

B - цветовая гамма от голубого до желтого

То есть двумя показателями в совокупности определяется цвет и одним показателем определяется его освещенность.

LAB - Это аппаратно-независимая цветовая модель, то есть она не зависит от способа передачи нам цвета. Она содержит в себе цвета как RGB так и CMYK, и grayscale, что позволяет ей с минимальными потерями конвертировать изображение из одной цветовой модели в другую.

Еще одним достоинством является то, что она, в отличие от цветовой модели HSB, соответствует особенностям восприятия цвета глазом человека.

Часто используется для улучшения качества изображения, и конвертирования изображений из одного цветового пространства в другое.

14.3.7 Цветовая модель *YIQ*

Эта модель используется в цветном ТВ, тесно связана с цветной растровой графикой и представляет собой вариант кодирования цветов *RGB,* обеспечивающий совместимость с черно − белым телевидением. Это пространство используется в ТВ системе NTSC в США. Координата *Y* при этом совпадает с координатой *Y* в колориметрической системе МКО (Международная комиссия по освещению- *Comission Internationale de l'Eclairage, CIE).* Компонент I представляет тон, а компонент *Q* - насыщенность. Преобразование модели *RGB* в модель *YIQ* выполняется в соответствии с системой уравнений:

$$\begin{bmatrix} Y \\ I \\ Q \end{bmatrix} = \begin{bmatrix} 0{,}299 & 0{,}587 & 0{,}114 \\ 0{,}596 & -0{,}275 & -0{,}321 \\ 0{,}212 & -0{,}523 & 0{,}311 \end{bmatrix} \times \begin{bmatrix} R \\ G \\ B \end{bmatrix}. \quad \textbf{(3.4)}$$

Соответственно обратное преобразование выполняется следующим образом:

$$\begin{bmatrix} R \\ G \\ B \end{bmatrix} = \begin{bmatrix} 1 & 0{,}956 & 0{,}620 \\ 1 & -0{,}272 & -0{,}647 \\ 1 & -1{,}108 & 1{,}705 \end{bmatrix} \times \begin{bmatrix} Y \\ I \\ Q \end{bmatrix}. \quad (3.5)$$

14.3.8. Цветовая модель *L*a*b** МКО 1976

Эта система координат обеспечивает относительно точное представление цветов в соответствии с системой цветов, разработанной в 1905 году художником Манселлом. Эта система может быть получена после преобразования системы *RGB* в цветовую координатную систему *XYZ* МКО 1931 в соответствии с уравнениями:

$$\begin{bmatrix} X \\ Y \\ Z \end{bmatrix} = \begin{bmatrix} 2{,}769 & 1{,}7518 & 1{,}13 \\ 1 & 4{,}5907 & 0{,}0601 \\ 0 & 0{,}0565 & 5{,}5943 \end{bmatrix} \times \begin{bmatrix} R \\ G \\ B \end{bmatrix}.$$

Сумма коэффициентов при компонентах составляет 5,651. С учетом нормировки преобразование следует выполнять в соответствии с системой:

$$\begin{bmatrix} X \\ Y \\ Z \end{bmatrix} = \begin{bmatrix} 0{,}49 & 0{,}31 & 0{,}1999646 \\ 0{,}17695983 & 0{,}81242258 & 0{,}0106175 \\ 0 & 0{,}01008 & 0{,}989913 \end{bmatrix} \times \begin{bmatrix} R \\ G \\ B \end{bmatrix}. \quad (3.6)$$

Затем выполняется преобразование системы *XYZ* в систему *L*a*b** в соответствии с уравнениями (3.7). Координата *L** определяет яркость цвета, *a** - соотношение красного и зеленого цветов, *b** - соотношение синего и зеленого.

$$L^* = \begin{cases} 116(Y/Y_0)^{1/3} - 16, & Y/Y_0 > 0{,}008856 \\ 903{,}3\,Y/Y_0, & \text{иначе} \end{cases}$$

$$a^* = 250\left(f\left((X/X_0)^{1/3}\right) - f\left((Y/Y_0)^{1/3}\right)\right)$$

$$b^* = 100\left(f\left((Y/Y_0)^{1/3}\right) - f\left((Z/Z_0)^{1/3}\right)\right),$$

$$\text{где } f\left(t^{1/3}\right) = \begin{cases} t^{1/3}, & t > 0{,}008856 \\ 7{,}787t + 16/116, & \text{иначе} \end{cases}, \quad \textbf{(3.7)}$$

X, Y, Z - координаты опорного белого цвета в системе*XYZ.*

14.3.9 Цветовая модель *L*H°C**

Цилиндрические координаты этого пространства соответствуют как эмпирической системе Манселла, так и согласуются с физиологической моделью цветного зрения. Эти координаты известны как психометрическая яркость, тон и насыщенность и задаются по формуле:

$$L^* = L^*$$

$$H^\circ = \operatorname{arctg}\left(b^*/a^*\right)$$

$$C^* = [(a^{*2} + b^{*2})]^{1/2}. \quad (3.8)$$

При таком описании элементом является круговой цилиндрический сегмент, выделенный на рисунке 3.4. Поверхности элементарного

объема формируются в соответствии с рисунком при заданных диапазонах изменения яркости и цветности. Горизонтальные срезы формируются при условии постоянной яркости, вертикальные срезы, проходящие через ахроматическую ось OL^*, получаются при постоянном тоне, а части цилиндрических поверхностей, концентрических относительно оси OL^*, формируются при постоянной насыщенности.

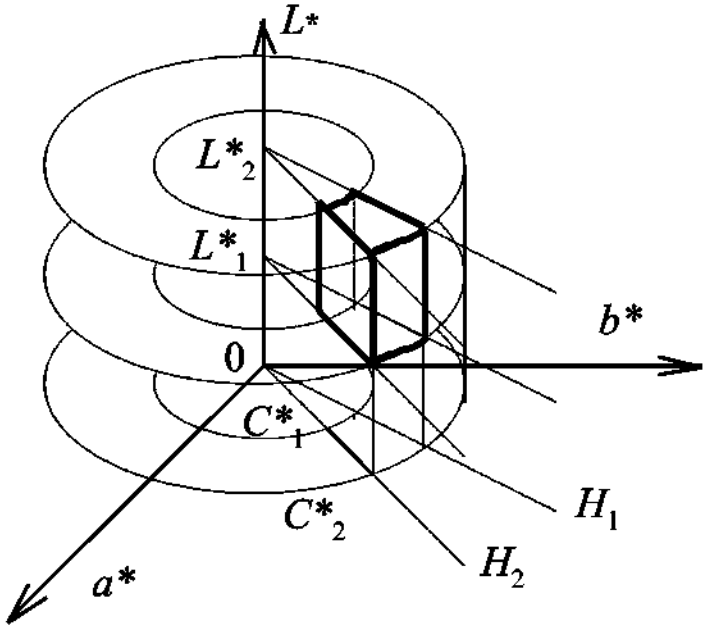


Рисунок 3.4 Элемент в цветовом координатном пространстве *L*a*b**. 3.6 Цветовая модель *HSI*

$$I = \frac{R+G+B}{3}. \quad (3.9)$$

$$S = 1 - \frac{3\min(R,G,B)}{R+G+B}. \quad (3.10)$$

$$H = \begin{cases} (G-B)/[3(R+G-2B)], & \text{если } B = \min(R,G,B) \\ (B-R)/[3(G+B-2R)]+1/3, & \text{если } R = \min(R,G,B) \\ (R-G)/[3(R+B-2G)]+2/3, & \text{если } G = \min(R,G,B) \end{cases}. \quad (3.11)$$

Выполним обратное преобразование цветового координатного пространства *HSI* в пространство *RGB* и получим: если $H \le 1/3$, то

$$\begin{cases} B = (1-S)I \\ G = 9SIH + B \\ R = 3I - (G+B) \end{cases}; \quad (3.12)$$

если $1/3 < H \le 2/3$, то

$$\begin{cases} R = (1-S)I \\ B = 9SI(H - 1/3) + R; \\ G = 3I - (R+B) \end{cases} \quad (3.13)$$

если $H > 2/3$, то

$$\begin{cases} G = (1-S)I \\ R = 9SI(H - 2/3) + G. \\ B = 3I - (R+G) \end{cases} \quad (3.14)$$

14.3.10 Цветовая модель *HLS*

Эти цветовые координаты введены Тененбаумом (Стэнфордский исследовательский институт) и широко используются при анализе сцен.

Тон и насыщенность определяются через *rgb* координаты, определяемые как нормированные тристимульные значения:

$$r = \frac{R}{R+G+B}; \; g = \frac{G}{R+G+B}; \; b = \frac{B}{R+G+B}. \quad (3.15)$$

Локус $r+g+b=1$ определяет треугольник Максвелла, изображенный на рисунках 3.5 и 3.6. На рисунке 3.5 приняты следующие обозначения: *P*-цветной элемент; W-серый, $r=g=b=1/3$; P'-пересечение *OP* с плоскостью треугольника. Пересечение вектора *OP* с плоскостью треугольника Максвелла определяет тон и насыщенность в соответствии с выражениями 3.16:

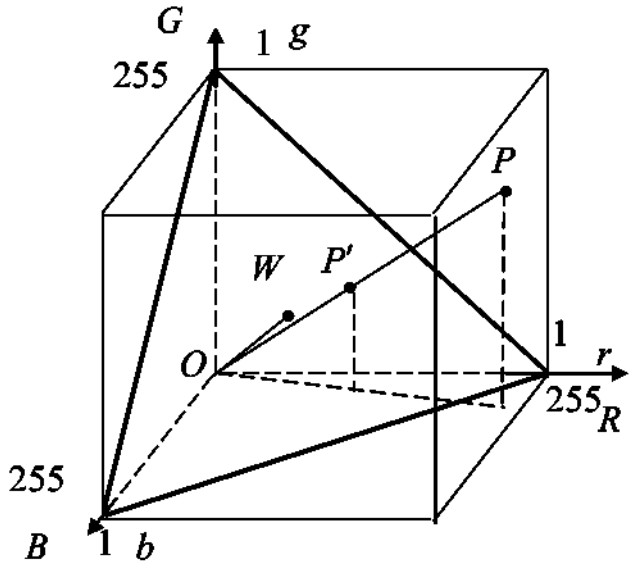


Рисунок 3.5 Цветовое координатное пространство *RGB.*

$$H=\phi\ (0\le\phi\le 2\pi);\quad (3.16\ \text{а})$$

$$S=WP'/WA\ (0\le S\le 1).\quad (3.16\ \text{б})$$

Яркость *L* пропорциональна длине вектора *OP* на рисунке 3.4 и определяется в соответствии с уравнением:

$$L=(R+G+B)/3.. \quad (3.17)$$

Нейтральная точка, или точка серого, *W* представляет точку с равными компонентами *R, G, B.* Относительно этой точки определяются координаты *Hu S* (в соответствии с рисунком 3.6).

Выполним прямое преобразование, чтобы затем получить формулы обратного преобразования из пространства *HLS* в пространство *RGB.*

Треугольник Максвелла задается тремя точками с координатами (1,0,0), (0,1,0) и (0,0,1) в координатной системе *rgb.* Уравнение плоскости, проходящей через эти точки, в соответствии с уравнением плоскости в отрезках, имеет вид: $\frac{r}{1}+\frac{g}{1}+\frac{b}{1}=1,$ откуда $r+g+b=1.$

(3.18)

Точка *W* является центром тяжести треугольника Максвелла и имеет координаты (1/3, 1/3, 1/3). Угол между *O W* и плоскостью треугольника Максвелла составляет 90°:

$$\arcsin\left(\frac{\frac{1}{3}+\frac{1}{3}+\frac{1}{3}}{\sqrt{3}\sqrt{\frac{1}{9}+\frac{1}{9}+\frac{1}{9}}}\right)=\arcsin(1)=\frac{\pi}{2},\quad (3.19)$$

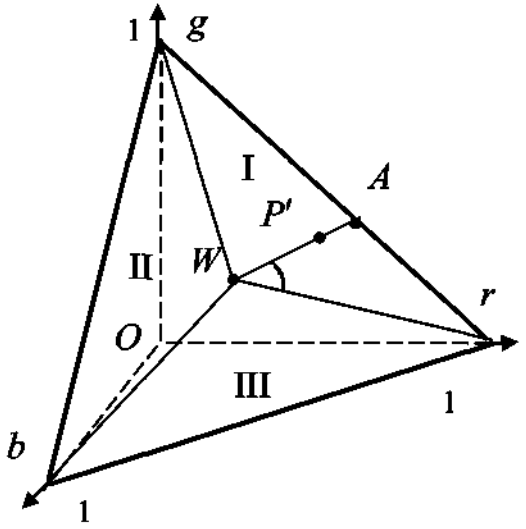


Рисунок 3.6 Цветовое координатное пространство *HLS.*

а модуль вектора *O W*определяется в соответствии с выражением:

$$|OW|=\sqrt{1/9+1/9+1/9}=\sqrt{3}/3.\quad (3.20)$$

Пусть точка P' имеет координаты *(r,g ,b)* , тогда уравнение прямой *OP* можно записать в виде :

$$\frac{r-r_O}{r_P-r_O}=\frac{g-g_O}{g_P-g_O}=\frac{b-b_O}{b_P-b_O},$$

$$r/r_1=g/g_1=b/b_1.\quad (3.21)$$

Отсюда направляющий вектор прямой *OP* имеет координаты *(r ,g ,b*). Определим угол между прямой *OP* и плоскостью треугольника
Максвелла:

$$\begin{cases} r+g+b-1=0 \\ r/r_1 = g/g_1 = b/b_1 \end{cases}. \quad (3.22)$$

$$\sin\gamma_3 = \frac{r_1+g_1+b_1}{\sqrt{3}\sqrt{r_1^2+g_1^2+b_1^2}} = \frac{1}{\sqrt{3}\sqrt{r_1^2+g_1^2+b_1^2}}. \quad (3.23)$$

$$WP' = OW\, ctg\,\gamma_3 = \frac{1}{\sqrt{3}}\frac{\sqrt{1-\sin^2\gamma_3}}{\sin\gamma_3}. \quad (3.24)$$

Уравнение прямой *gr,* где *g(0,1,0), r(1,0,0)* имеет вид:

$$\frac{r}{1} = \frac{g-1}{-1}, \text{ или } r=1-g. \quad (3.25)$$

Координаты точки P' определяются как координаты точки пересечения плоскости треугольника Максвелла и прямой *OP:*

$$\begin{cases} b+r+g-1=0 \\ r/r_1 = g/g_1 \\ r/r_1 = b/b_1 \end{cases}, \quad (3.26)$$

$$(b_1/g_1)g + (r_1/g_1)g + g - 1 = 0, \quad (3.27)$$

$$g = \frac{1}{b_1/g_1 + r_1/g_1 + 1} = \frac{g_1}{b_1+g_1+r_1} = g_1, \quad (3.28)$$

аналогично $b = b$, $r = r$. То есть координаты точек *P* и P' совпадают.
Уравнение прямой
$WP'\ (W(1/3,\ 1/3,\ 1/3), P'\ (b_1, g_1, r_1))$ имеет вид:

$$\left(r-\frac{1}{3}\right)\Big/\left(r_1-\frac{1}{3}\right) = \left(g-\frac{1}{3}\right)\Big/\left(g_1-\frac{1}{3}\right) = \left(b-\frac{1}{3}\right)\Big/\left(b_1-\frac{1}{3}\right). \quad (3.29)$$

Рассмотрим 3 случая: первый, когда точка P' находится в секторе I, в треугольнике *RWG,* θ =0°; второй, когда точка P' находится в секторе II, в треугольнике *GWB,* θ =120°; третий, когда точка P' находится в секторе III, в треугольнике *BWR,* θ_0 =240°.
Рассмотрим сектор I. Координаты точки *A* определяются как координаты точки пересечения прямых WP' и *GR.* Прямая *GR* задается системой:

$$\begin{cases} b=0 \\ r=1-g \end{cases}. \quad (3.30)$$

Из (3.29) при *b=0* получим:

$$\frac{-\frac{1}{3}}{b_1-\frac{1}{3}} = \frac{g-\frac{1}{3}}{g_1-\frac{1}{3}}; g = \frac{b_1-g_1}{3b_1-1} = \frac{b_1-g_1}{2b_1-r_1-g_1}; r = 1-g = \frac{b_1-r_1}{2b_1-r_1-g_1}. \quad (3.31)$$

Из (3.31) координаты точки *A* задаются следующими значениями:

$$\begin{aligned} b_A &= 0 \\ r_A &= (b_1-r_1)/(2b_1-r_1-g_1) \\ g_A &= (b_1-g_1)/(2b_1-r_1-g_1) \end{aligned}. \quad (3.32)$$

$$|WA| = \sqrt{\left(-\frac{1}{3}\right)^2 + \left(\frac{b_1-r_1}{2b_1-r_1-g_1}-\frac{1}{3}\right)^2 + \left(\frac{b_1-g_1}{2b_1-r_1-g_1}-\frac{1}{3}\right)^2} = \frac{\sqrt{(1-3b_1)^2+(1-3r_1)^2+(1-3g_1)^2}}{3|2b_1-r_1-g_1|}. \quad (3.33)$$

$$|WP'| = \sqrt{\left(r_1-\frac{1}{3}\right)^2 + \left(g_1-\frac{1}{3}\right)^2 + \left(b_1-\frac{1}{3}\right)^2} = \frac{1}{3}\sqrt{(3r_1-1)^2+(3g_1-1)^2+(3b_1-1)^2}. \quad (3.34)$$

Насыщенность, задаваемая как отношение модулей WP' и WA, вычисляется делением (3.34) на (3.33):

$$S = \frac{|WP'|}{|WA|} = |2b_1 - r_1 - g_1| = |3b_1 - 1| = |1 - 3b_1|. \quad (3.35)$$

Для определения тона необходимо вычислить угол между прямой WR и прямой WP':

$$\cos\phi = \frac{(r_{P'} - r_W)(r_R - r_W) + (g_{P'} - g_W)(g_R - g_W) + (b_{P'} - b_W)(b_R - b_W)}{|WR| \times |WP'|}$$

$$= \frac{\left(r_1 - \frac{1}{3}\right)\left(1 - \frac{1}{3}\right) + \left(g_1 - \frac{1}{3}\right)\left(-\frac{1}{3}\right) + \left(b_1 - \frac{1}{3}\right)\left(-\frac{1}{3}\right)}{\sqrt{\left(r_1 - \frac{1}{3}\right)^2 + \left(g_1 - \frac{1}{3}\right)^2 + \left(b_1 - \frac{1}{3}\right)^2} \times \sqrt{\left(\frac{2}{3}\right)^2 + \left(\frac{1}{3}\right)^2 + \left(\frac{1}{3}\right)^2}} =$$

$$= \frac{2r_1 - g_1 - b_1}{\sqrt{6\left[\left(r_1 - \frac{1}{3}\right)^2 + \left(g_1 - \frac{1}{3}\right)^2 + \left(b_1 - \frac{1}{3}\right)^2\right]}}.$$

Для сектора II θ =120°, производя вычисления, аналогичные выполненным для сектора I, с учетом того, что точка A определяется как точка пересечения прямых WP' и GB, а тон задается углом между прямыми WP' и GB плюс начальное смещение θ = 120°, получим следующие выражения:

$$S = |2r_1 - g_1 - b_1|, \quad (3.37)$$

$$\cos\phi = \frac{2g_1 - r_1 - b_1}{\sqrt{6\left[\left(r_1 - \frac{1}{3}\right)^2 + \left(g_1 - \frac{1}{3}\right)^2 + \left(b_1 - \frac{1}{3}\right)^2\right]}}. \quad (3.38)$$

Для сектора III θ =240°, производя вычисления, аналогичные

выполненным для сектора I, с учетом того, что точка A определяется как точка пересечения прямых WP' и BR, а тон задается углом между прямыми WP' и BR плюс начальное смещение θ =240°, получим следующие выражения:

$$S = |2g_1 - r_1 - b_1|, \quad (3.39)$$

$$\cos\phi = \frac{2b_1 - r_1 - g_1}{\sqrt{6\left[\left(r_1 - \frac{1}{3}\right)^2 + \left(g_1 - \frac{1}{3}\right)^2 + \left(b_1 - \frac{1}{3}\right)^2\right]}}. \quad (3.40)$$

Обобщая (3.35)÷(3.40), можно записать

$$H = \theta_0 + \arccos\left[N \Big/ \sqrt{6\left[\left(r - \frac{1}{3}\right)^2 + \left(g - \frac{1}{3}\right)^2 + \left(b - \frac{1}{3}\right)^2\right]}\right], \quad (3.41)$$

где

N=2r-g-b, 2g-b-r, 2b-r-g и $\theta_0 = 0°,\ 120°,\ 240°$

в секторах I, II, III соответственно.

$$S=1-3\min(r,g,b). \quad (3.42)$$

Обратное преобразование, так же как и прямое, будем выполнять для каждого сектора отдельно.

В секторе I при θ=0° исходные данные представлены системой (3.43):

$$\begin{cases} r + g + b = 1 \\ H = \arccos\left[(3r-1) \Big/ \sqrt{6\left[\left(r - \frac{1}{3}\right)^2 + \left(g - \frac{1}{3}\right)^2 + \left(b - \frac{1}{3}\right)^2\right]}\right]. \\ S = |2b - r - g| = |3b - 1| = 1 - 3b \end{cases} \quad (3.43)$$

Обозначим

$$x = 3r - 1. \quad (3.44)$$

$$b = -\frac{S}{3} + \frac{1}{3},$$

$$g = 1 - r - b = \frac{2}{3} - r + \frac{S}{3}. \quad (3.45)$$

Отсюда $3g - 1 = S - x,\ 3b - 1 = -S.$

$$\cos\phi = x \Big/ \sqrt{\frac{6}{9}\left(x^2 + (S-x)^2 + S^2\right)}. \quad (3.46)$$

Возведя правую и левую части уравнения (3.46) в квадрат, получим:

$$\frac{4}{3}\cos^2\phi\left(x^2 + S^2 - Sx\right) = x^2. \quad (3.47)$$

Корни уравнения (3.47):

$$x_{1,2} = 2S\frac{\cos^2\phi \pm \sqrt{3}|\cos\phi|\sin\phi}{\cos^2\phi - 3\sin^2\phi}. \quad (3.48)$$

При $\phi \in \left[0, \frac{\pi}{2}\right]$ из (3.48) получим:

$$x_1 = S\frac{\cos\phi}{\cos\phi + \cos(120° - \phi)} = S\frac{\cos\phi}{\cos(60° - \phi)}, \quad (3.49\ а)$$

$$x_2 = -S\frac{\cos\phi}{\cos(120° - \phi)}. \quad (3.49\ б)$$

Такую же пару значений x мы получим при раскрытии модуля в случае отрицательных значений косинуса при $\phi \in \left[\frac{\pi}{2}, \frac{2\pi}{3}\right]$, только x_1 и x_2 при этом поменяются местами. Поскольку область изменения функции, задаваемой уравнением (3.49 б), не удовлетворяет геометрическому смыслу задачи, то следует этот корень считать посторонним. Решением является корень x_1.

В соответствии с (3.49 а) из (3.44) выражение для *r* имеет вид:

$$r = \frac{1}{3} + \frac{S\cos\phi}{3[\cos(60° - \phi)]}. \quad (3.50)$$

В соответствии с (3.49 а) из (3.45) выражение для *g* имеет вид:

$$g = \frac{1+S}{3} - \frac{S\cos\phi}{3[\cos(60° - \phi)]}. \quad (3.51)$$

Выполнив аналогичные вычисления для секторов II, в котором θ_0 =120°, и III, в котором θ_0 =240°, и обобщив полученные решения, получим следующие уравнения для обратного преобразования из пространства *HLS* в пространство *RGB:*

$$\begin{cases} x_1 = (1-S)/3; \\ x_2 = S\cos\phi/[3\cos(60° - \phi)] + 1/3, \\ x_3 = 1 - (x_1 + x_2) \end{cases} \quad (3.52)$$

где $\phi = H,\ r = x_2,\ g = x_3,\ b = x_1$, при *H*< 120°;

иначе если *H*≤240°, то $\phi = H$-120°, r= x_1, g= x_2, b= x_3 , иначе если *H*>240°, то $\phi = H$-240°, $r = x_3,\ g = x_1,\ b = x_2$.

$$R = 3Lr;\ G = 3Lg;\ B = 3Lb. \quad (3.53)$$

Преобразование пространства *RGB* в пространство *HLS* выполняется в соответствии с уравнениями (3.52), (3.53).

14.3.11 Цветовая модель *L*u*v** МКО 1973

Эта модель равноконтрастного цветового пространства отличается от * * * пространства *Lab* цветовыми координатами *u*v** (яркости в этих пространствах совпадают). Если *(a*b*)* являются нелинейным преобразованием *(X,Y)* МКО, то (U*V*) связаны с *(X,Y)* линейным преобразованием.

$$\begin{aligned} L^* &= 25(100Y/Y_0)^{1/3} - 16,\ 1 <= Y <= 100) \\ u^* &= 13L^*(u' - u_0') \\ v^* &= 13L^*(v' - v_0') \end{aligned}, \quad (3.54)$$

где

$$u' = \frac{4X}{X + 15Y + 3Z};\qquad v' = \frac{9Y}{X + 15Y + 3Z}; \quad (3.55)$$

$$u'_0 = \frac{4X_0}{X_0 + 15Y_0 + 3Z_0}; \qquad v'_0 = \frac{9Y_0}{X_0 + 15Y_0 + 3Z_0}.$$

14.3.12. Метрическое векторное цветовое пространство

На оппонентной теории цвета разработан подход к представлению цветов в линейных векторных пространствах**. В отличие от аффинного пространства, где отсутствуют определения угла и расстояния, полученное пространство является метрическим векторным цветовым пространством.** Оно образуется откликами зрительной системы, модули векторов которых равны яркостным амплитудам. Реальные цвета образуют конус в метрическом векторном пространстве, положение векторов в котором зависит от соотношения яркостных амплитуд.

В цветовой фотометрии определяются основные величины для количественного описания цветов: амплитуда цвета, цветовая яркость, насыщенность и цветовой тон.

Цветовая амплитуда равна векторной сумме амплитуд основных откликов зрительной системы: $\mathbf{A}_c = \mathbf{A}_r + \mathbf{A}_g + \mathbf{A}_b$ и может быть выражена через модули цветовых векторов в скалярном виде:

$$A_c = \sqrt{A_r^2 + A_g^2 + A_b^2 + 2A_rA_g\cos\alpha_{rg} + 2A_gA_b\cos\alpha_{gb} + 2A_bA_r\cos\alpha_{br}}, \quad (3.56)$$

где α_{rg} , α_{gb} , α_{br} - углы между векторами откликов в метрическом векторном пространстве (в соответствии с рисунком 3.7).

Цветовая яркость $L_c = Ac$ может быть выражена через цветовые амплитуды и яркости L_r, L_g, L_b в соответствии с уравнением:

$$L_c = L_r + L_g + L_b + 2A_rA_g\cos\alpha_{rg} + 2A_gA_b\cos\alpha_{gb} + 2A_bA_r\cos\alpha_{br}. \quad (3.57)$$

Белому равноэнергетическому цвету в метрическом векторном цветовом пространстве соответствует вектор белого AW. Плоскость цветности перпендикулярна вектору равноэнергетического белого *AW*.

Насыщенность цвета в метрическом векторном пространстве количественно определяется косинусом угла между вектором данного цвета и вектором равноэнергетического белого цвета: S = cosO. (3.58)

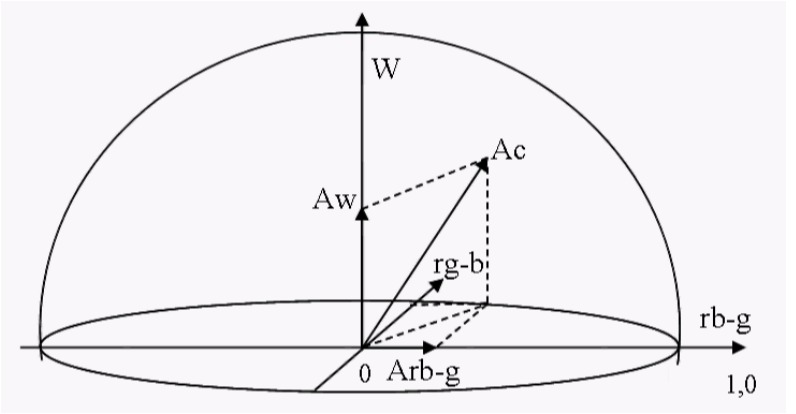


Рисунок 3.7 Представление цветов в метрическом векторном пространстве.

Цветовой тон определяется углом φ в плоскости цветности между составляющей вектора цвета в плоскости цветности $A_{цс}$ и одним из векторов оппонентных сигналов цветности, например, вектором $\mathbf{A}_{rb\text{-}g}$. Численно цветовой тон определяется выражением:

$$\varphi = (-1)^k \arccos(A_{rb\text{-}g}/A_c) + n\pi, \quad (3.59)$$

где постоянные *n* и *k* зависят от знака оппонентных сигналов *rb-g* и *rg-b*.

Ожидается, что построение ТВ систем на принципе постоянной цветовой яркости позволит приблизить воспроизведение цветных изображений к фотометрически точным условиям цветопередачи, повысить четкость на цветных деталях изображений, в том числе в насыщенных цветах, приблизив ее к четкости черно-белых изображений. Эта теория находится в развитии. Проблема состоит в определении основных величин. В колориметрии цвет численно определяется через функции относительного спектрального распределения реакций глаза, обусловленных работой колбочек, в неметрическом векторном пространстве. Предлагаемое метрическое векторное цветовое пространство позволит определить цветовую фотометрию, основанную на откликах цветовых зрительных каналов. Это пространство равномерно и в нем определены скалярные умножения. Основным законом цветовой фотометрии является аддитивность векторов. Предложена модель цветового зрения. Показано, что цветовые отклики могут быть описаны векторами в метрическом пространстве. В соответствии с предложенной моделью,

основными величинами цветовой фотометрии являются амплитуды цветовых реакций и углы между векторами цветов. Гетерохромная яркость зависит от амплитуды хроматического отклика, а тон и насыщенность цвета зависят от углов.

ПРИЛОЖЕНИЕ 1

При изучении многомерных СП, т. е. случайных функций многих переменных, приходится оперировать с многомерными массивами данных. Удобным математическим аппаратом для такого анализа являются многомерные матрицы и тензоры, элементы теории которых изложены в первой части этого приложения в удобной для применения форме.

При решении различных задач обработки изображений и в других приложениях нередко требуется возводить в большие степени и обращать различные матрицы. Во второй части приложения приведены сведения из изящной теории матричных функций, применение которых позволяет выполнить эти операции с небольшими вычислительными затратами.

П.1.1. Многомерные матрицы и тензоры

Рассмотрим способы задания линейных преобразований скаляров, векторов и многомерных массивов. Напомним, что преобразование y=y(x) линейного пространства элементов x в линейное пространство элементов y называется линейным, если y(cx) = cy(x) и $y(x+x_1) = y(x) + y(x_1)$ для любого скаляра c и любых x и x_1.

Любое линейное преобразование скалярных величин x в скалярные величины y может быть представлено в форме $y = a_1x$, где a1 – скаляр. Это соотношение запишем в виде $y_1 = a_1x_1$.

Для задания преобразования векторов $\bar{x} = (x_1,\dots,x_m)^T$ в векторы $\bar{y} = (y_1,\dots,y_n)^T$ необходимо задать зависимость каждой из компонент вектора $\bar{y}$ от компонент вектора $\bar{x}$. В случае линейности преобразования эта зависимость имеет вид

$$y_i = \sum_{j=1}^{m} a_{ij} x_j, i = 1,2,\ldots, n.$$ (П.1)

Здесь a_{ij} – постоянные коэффициенты, которые представим в виде матрицы A. В матричных обозначениях (П.1) может быть записано как

$$\bar{y} - A\bar{x}$$. (П.2)

Рассмотрим теперь преобразование $m \times n$-матриц $X = \{x_{ij}\}$ в $p \times q$-матрицы $Y = \{y_{ij}\}$. Для линейности преобразования нужно, чтобы компоненты матрицы Y линейно зависели от компонент X:

$$y_{ij} = \sum_{k=1}^{m} \sum_{l=1}^{n} a_{ijkl} x_{kl}, i = 1,\ldots, p, j = 1,\ldots, q$$,

где $a_{ij}k_l$ – постоянные коэффициенты.

Обобщая эти частные случаи, можно сделать вывод, что любое линейное преобразование m-мерных массивов $X = \{x_{i_1 i_2 \ldots i_m}\}$ в n–мерные массивы $Y = \{y_{i_1 i_2 \ldots i_n}\}$ может быть задано равенством

$$y_{j_1 j_2 \ldots j_n} = \sum_{i_1} \cdots \sum_{i_m} a_{j_1 \ldots j_n i_1 \ldots i_m} x_{i_1 \ldots i_m}$$, (П.3)

где $a_{j_1 \ldots j_n i_1 \ldots i_m}$ – постоянные для данного преобразования коэффициенты. Обозначая совокупность этих коэффициентов через A, можно, по аналогии с (П.2), представить (П.3) в краткой форме Y=AX при очевидных соглашениях о смысле этой записи. Итак, для задания линейного преобразования массива размерности m в массив

размерности n необходимо задать массив коэффициентов размерности m+n.

Матрицей размерности n называется совокупность действительных или комплексных чисел

$$A = \{a_{i_1 i_2 \ldots i_n} : i_1 = 1,\ldots, M_1,\ldots, i_n = 1,\ldots, M_n\}$$.

При этом n называется размерностью матрицы A, а $M_1 \times M_2 \times \ldots \times M_n$ – ее размерами. При n=1 и M_1=1 получаем скаляр; при n=1 и M_1=m – вектор; при n=2 – обычную (двумерную) $M_1 \times M_2$-матрицу $\{a_{i_1 i_2}\}$, которую представляют в виде таблицы чисел (рис. П.1).

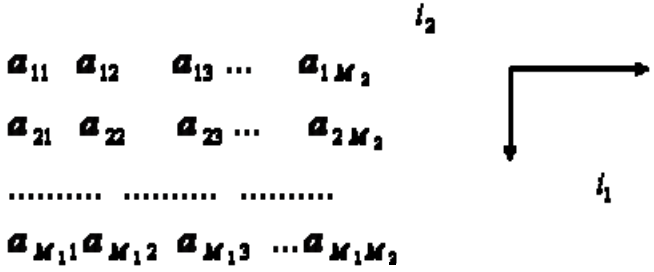


Рис. П.1.

Трехмерную матрицу $A = \{a_{i_1 i_2 i_3}\}$ можно представить как трехмерную таблицу чисел, а графически изобразить "послойно", обозначая направления изменения соответствующих индексов. Например, на рис. П.2 изображена трехмерная матрица размеров $4 \times 3 \times 2$. В трехмерном пространстве две таблицы, разделенные на рисунке пунктиром, находились бы одна за другой.

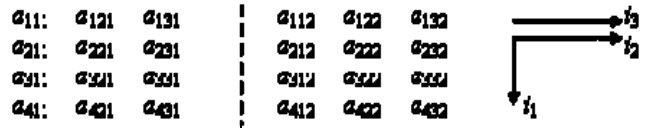


Рис. П.2.

На рис. П.3 изображена четырехмерная матрица $B = \{b_{i_1 i_2 i_3 i_4}\}$ размеров $2 \times 3 \times 2 \times 2$. Подобным образом можно изобразить графически матрицу любой размерности.

Если в n-мерной матрице A фиксировать какие-нибудь m ее индексов, а остальные оставить изменяющимися, то получится (n-m)-мерная матрица, называемая **сечением матрицы A**. Например, матрица справа от пунктира на рис. П.2 является (i_3=2)-сечением, а вторая строка слева от пунктира – (i_1=2, i_3=1)-сечением матрицы A.

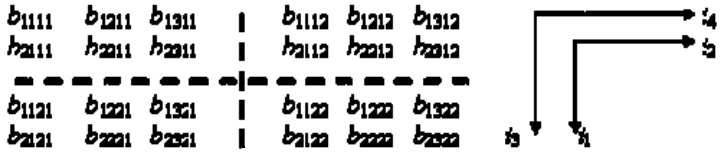


Рис. П.3.

Умножение матрицы на число и сумма (разность) матриц одинаковых размеров определяются так же, как и для двумерных матриц.

Умножение многомерных матриц аналогично умножению двумерных матриц, но более разнообразно и выполняется с применением правил тензорного исчисления **зацепления, свертывания или сокращения индексов: если некоторый индекс встречается в выражении дважды, то выражение должно быть по этому индексу просуммировано.** Например,

$$a_i b_i = \sum_i a_i b_i, \quad a_{ij} x_j = \sum_j a_{ij} x_j, \quad a_{ij} x_i = \sum_i a_{ij} x_i .$$

Индексы, по которым производится суммирование, называются **немыми**. Индексы, не являющиеся немыми, называются **свободными.** После выполнения суммирования по немому индексу i данный индекс в результирующем выражении исчезает, поэтому немой индекс может быть заменен любым другим, не встречающимся в выражении, например, $a_{ij}x_j=a_ikxk$. Свободные же индексы в результирующем выражении сохраняются.

Применяя описанное правило, можно получать различные произведения одних и тех же матриц, варьируя совокупность немых индексов. Например, для матриц A и B, изображенных на рис. П.1 и рис. П.2, можно образовать произведения:

$$F = \{f_{i_1 i_2 i_3 j_1 j_2 j_3 j_4}\} = \{a_{i_1 i_2 i_3}\}\{b_{j_1 j_2 j_3 j_4}\} = \{a_{i_1 i_2 i_3} b_{j_1 j_2 j_3 j_4}\}, \quad (П.4)$$

$$G = \{g_{i_1 i_2 j_1 j_2 j_3}\} = \{a_{i_1 i_2 k}\}\{b_{j_1 j_2 j_3 k}\} = \{\sum_{k=1}^{2} a_{i_1 i_2 k} b_{j_1 j_2 j_3 k}\}, \quad (П.5)$$

$$H = \{h_{i_1 j_2 j_4}\} = \{a_{i_1 k}\}\{b_{k j_2 j_4}\} = \{\sum_{k=1}^{3} \sum_{l=1}^{2} a_{i_1 k} b_{k j_2 j_4}\} \quad (П.6)$$

и т. д.

При использовании формулы (П.4) умножение трехмерной матрицы A на четырехмерную B приводит к семимерной матрице F, получающейся умножением каждого элемента матрицы A на каждый элемент матрицы B. Такое произведение называется **прямым или внешним**, все индексы в нем свободные, обозначается оно $A \times B$.

В (П.5) произведением тех же матриц является пятимерная матрица G, так как третий индекс матрицы A и третий индекс матрицы B обозначены одним символом k, т. е. являются немыми и после суммирования исчезают. Например, элемент $g_{42121} = a_{42k}b_{12k1} = a_{421}b_{1211} + a_{422}b_{1221}$.

В (П.6) результатом умножения является трехмерная матрица, так как немых индексов уже два – k и l. Например, элемент

$$h_{321} = a_{3kl}b_{k21} = \sum_{k=1}^{3}\sum_{l=1}^{2} a_{3kl}b_{k21}$$

.

Произведения, в которых имеются немые индексы, называются (в отличие от внешних произведений) **внутренними** и могут быть выполнены в два этапа: сначала выполняется прямое произведение, а затем свертываются нужные пары индексов, т. е. заменяются одним. Как и при умножении двумерных матриц, необходимо соответствие размеров. Например, для выполнимости (П.6) необходимо, чтобы размеры $M_1 \times M_2 \times M_3$ и $N_1 \times N_2 \times N_3 \times N_4$ матриц A и B удовлетворяли условиям: $M_2 = N_2$ и $M_3 = N_1$. Свертывание по каждому индексу уменьшает размерность прямого произведения на два. Следовательно, при умножении двух матриц размерностей m и n можно получить матрицы размерностей m+n, m+n-2, …, |m-n|. Если m=n, то результатом умножения может быть даже скаляр, как, например, в произведении $\{a_{ij}\}\{b_{ji}\}$.

В этих обозначениях обычное умножение двумерных матриц записывается в виде $\{a_{ik}\}\{b_{kj}\}$, умножение матрицы на вектор – $\{a_{ij}\}\{x_j\}$, скалярное произведение векторов – $\{x_i\}\{y_i\}$. При этом отпадает необходимость в символе операции транспонирования: $\bar{x}^T A^T$ записывается как $\{a_{ij}\}\{x_i\}$.

Если в многомерной матрице $A = \{a_{i_1 i_2 \ldots i_n}\}$ переставить местами индексы, то получится, вообще говоря, другая матрица, называемая **изомерой** матрицы A. Количество изомер равно n!

Например, если в $4 \times 3 \times 2$-матрице $A = \{a_{i_1 i_2 i_3}\}$ (рис. П.1) переставить местами i_2 и i_3, то получится $4 \times 2 \times 3$-матрица, показанная на рис. П.4. Такие операции являются обобщением операции транспонирования двумерных матриц. Если все изомеры совпадают между собой, то матрица A называется **симметричной**.

Естественным образом определяется произведение трех и более

$$\{a_{ij}\}\{b_{jkl}\}\{c_{lm}\} = \{\sum_{j,l} a_{ij}b_{jkl}c_{lm}\} = \{d_{ikm}\}$$

матриц, например, .

При этом операции сложения матриц, умножения их на число и перемножения обладают, как и в случае двумерных матриц, свойствами ассоциативности, коммутативности и дистрибутивности, кроме коммутативности умножения матриц, которое выполняется только с точностью до изомеры.

a_{111}	a_{112}	a_{121}	a_{122}	a_{131}	a_{132}
a_{211}	a_{212}	a_{221}	a_{222}	a_{231}	a_{232}
a_{311}	a_{312}	a_{321}	a_{322}	a_{331}	a_{332}
a_{411}	a_{412}	a_{421}	a_{422}	a_{431}	a_{432}

Рис. П.4.

Иногда удобно группировать индексы многомерных матриц, например, обозначить матрицу $\{a_{i_1 \ldots i_m j_1 \ldots j_n}\}$ как $\{a_{\bar{i}\bar{j}}\}$, где $\bar{i} = (i_1, \ldots, i_m)$ и $\bar{j} = (j_1, \ldots, j_n)$. При этом **групповой индекс** во всех операциях над матрицами выступает как единое целое, и индексы, входящие в $\bar{i}$ и $\bar{j}$, не могут быть переставлены. Матрица $\{a_{\bar{i}\bar{j}}\}$ имеет только две изомеры: $\{a_{\bar{i}\bar{j}}\}$ и

$\{a_{jI}\} = \{a_{j_1,\dots,j_n i_1,\dots,i_m}\}$, а не (m+n)! , и при m=n может оказаться симметричной, даже если $\{a_{i_1,\dots,i_m;i_1,\dots,i_n}\}$ не симметрична.

Как уже отмечалось, каждое линейное преобразование m-мерных массивов $X = \{x_{j_1\dots j_m}\}$ в n-мерные массивы $Y = \{y_{i_1\dots i_n}\}$ может быть записано в виде

$$Y = AX = \{a_{Ij}\}\{x_j\}, \qquad (П.7)$$

где $X = \{x_j\}$ и $Y = \{y_I\}$. И наоборот, каждая m+n-мерная матрица определяет некоторое линейное преобразование $\{x_j\} \to \{y_I\}$ по формуле (П.7). Если, кроме этого, Z=BY – линейное преобразование $\{y_I\} \to \{z_K\}$, то суперпозиция этих двух преобразований Z=B(AX)=(BA)X имеет матрицу $C = BA = \{b_{KI}\}\{a_{Ij}\}$.

Если базис, т. е. систему координат линейного пространства, сменить, то это приведет и к изменению матрицы данного преобразования, т. е. каждому линейному преобразованию будет соответствовать совокупность многомерных матриц. В тензорном анализе n-мерная матрица называется **тензором ранга n**, если выполнены некоторые условия, связанные с изменением элементов этой матрицы при замене системы координат. Например, выражение «тензор линейного преобразования» означает матрицу этого преобразования в соответствующей системе координат.

В рамках задач данной работы не требуется преобразований различных координат, поэтому упомянутые условия выполнены и справедлива

тензорная терминология. Например, если $\{x_j\}$ – СП и $y_I = \sum_j a_{Ij}x_j$, то будем называть $A = \{a_{Ij}\}$ тензором преобразования СП $\{x_j\}$ в СП $\{y_I\}$. Здесь же отметим, что применительно к СП удобнее записывать индексы компонент тензоров только снизу, используя верхний индекс как временной.

Особое значение имеют многомерные тензоры четной размерности 2n вида $A = \{a_{IJ}\}$, где групповые индексы $\bar{I} = (i_1,\dots,i_n)$ и $\bar{J} = (j_1,\dots,j_n)$ имеют одинаковые размеры $M_1 \times M_2 \times \dots \times M_n$. Такие тензоры называются **квадратными**.

Они задают линейные преобразования массивов $x_{\bar{J}}$ в массивы $y_{\bar{I}} = \sum_{\bar{J}} a_{\bar{I}\bar{J}}x_{\bar{J}}$ тех же размеров. Это сближает многомерные квадратные матрицы с квадратными двумерными.

Единичной матрицей называется квадратная матрица $E = \{\delta(\bar{i},\bar{j})\}$, где δ – символ Кронекера. Она задает тождественное преобразование EX=X. Матрицей, **обратной** к квадратной матрице $A = \{a_{IJ}\}$, называется матрица $A^{-1} = \{b_{IJ}\}$ такая, что $AA^{-1} = \{a_{IK}\}\{b_{KJ}\} = E$ и $A^{-1}A = \{b_{IK}\}\{a_{KJ}\} = E$.

Обратную матрицу можно вычислить следующим образом.

Пронумеруем все значения группового индекса $\bar{i}$ числами $N(\bar{i})$

от 1 до $M_1 M_2 .. M_n$ и составим двумерную матрицу $\overline{A} = \{\overline{a}_{N(I),N(J)}\}$, где $\overline{a}_{N(I),N(J)} = a_{IJ}$, которая называется **разверткой** матрицы А и определяется с точностью до одновременной перестановки строк и столбцов. **Детерминантом** матрицы А будем называть детерминант ее развертки: $\det(A) = \det(\overline{A})$. Очевидно, что детерминант не зависит от способа развертки. Если $\det(A) \neq 0$, то матрица А называется **неособенной**, а ее развертка имеет обратную матрицу $\overline{A}^{-1}$. Применяя к $\overline{A}^{-1}$ процедуру, обратную развертке, получим искомую обратную матрицу A^{-1}.

Рассмотрим теперь операции **дифференцирования тензоров**, составленных из функций.

Если имеется скалярная функция а(х) одного скалярного переменного

$$a'_x = \frac{da}{dx} = a'(x)$$

х, то ее производная определяется обычным способом. Если $\overline{a}(x) = (a_1(x),...a_n(x))^T$ – векторная функция скалярного аргумента х, то ее **производная** по х определяется как $\frac{d\overline{a}(x)}{dx} = \left(\frac{da_1(x)}{dx},...,\frac{da_n(x)}{dx}\right)^T$, т. е. следует взять производную по х от каждой из компонент вектора $\overline{a}(x)$.

Производная тензора $A(x) = \{a_{i_1 i_2,...,i_n}(x)\}$, зависящего от одного скалярного переменного х, определяется аналогично:

$$\frac{dA(x)}{dx} = \left\{\frac{da_{i_1 i_2,...,i_n}(x)}{dx}\right\}$$

, т. е. тензор дифференцируется поэлементно.

Пусть $a(\overline{x}) = a((x_1, x_2,..., x_n)^T)$ – скалярная функция векторного аргумента $\overline{x}$. Ее можно рассматривать и как функцию n скалярных переменных $x_1,...x_n$: $a(\overline{x}) = a(x_1,...x_n)$. Она имеет n частных производных $\frac{\partial a}{\partial x_1}, \frac{\partial a}{\partial x_2},..., \frac{\partial a}{\partial x_n}$, из которых можно составить вектор $\frac{da(\overline{x})}{d\overline{x}} = \left(\frac{\partial a(x_1,...,x_n)}{\partial x_1},...,\frac{\partial a(x_1,...,x_n)}{\partial x_n}\right)^T$, называемый **производной функции** $a(\overline{x})$ **по векторной переменной** $\overline{x}$. В общем случае скалярной функции $a(X) = a(\{x_{i_1...i_n}\})$ тензорного аргумента $X = \{x_{i_1...i_n}\}$ ее **производная** определяется равенством $\frac{da(X)}{dX} = \left\{\frac{\partial a(\{x_{j_1...j_n}\})}{\partial x_{i_1 i_2...i_n}}\right\}$ и является тензором той же размерности, что и Х.

Пусть $\overline{y}(\overline{x}) = (y_1(x_1...x_n),..., y_m(x_1...x_n))^T$ – векторная функция векторного аргумента $\overline{x}$. **Производной функции** $\overline{y}(\overline{x})$ **по вектору** $\overline{x}$ называется матрица

$$\frac{\partial y(x)}{dx} = \begin{pmatrix} \frac{\partial y_1}{\partial x_1} & \frac{\partial y_1}{\partial x_2} & \cdots & \frac{\partial y_1}{\partial x_n} \\ \cdots & \cdots & \cdots & \cdots \\ \frac{\partial y_m}{\partial x_1} & \frac{\partial y_m}{\partial x_2} & \cdots & \frac{\partial y_m}{\partial x_n} \end{pmatrix},$$

составленная из <u>частных производных</u> всех компонент векторной функции $Y(X)$ по всем компонентам ее аргумента, что можно записать в сокращенном виде, как

$$\frac{\partial Y(X)}{dX} = \left\{ \frac{\partial y_i}{\partial x_j} \right\}. \qquad (П.8)$$

По аналогии с (П.8), **<u>производная</u> тензора** $Y = \{y_{i_1 \dots i_n}\}$ **ранга n по тензору** $X = \{x_{j_1 \dots j_m}\}$ **ранга m** определяется как

$$\frac{dY}{dX} = \left\{ \frac{\partial y_{i_1 \dots i_n}}{\partial x_{j_1 \dots j_m}} \right\} = \left\{ \frac{\partial y_i}{\partial x_j} \right\} \qquad (П.9)$$

и является тензором ранга n+m. Очевидно, что dX/dX = E – единичный тензор.

Рассмотрим тензор $Y = Y(X) = \{y_i(\{x_j\})\}$ и дадим его аргументу $X = \{x_j\}$ приращение $\Delta X = \{\Delta x_j\}$, т. е. дадим каждому аргументу x_j приращение Δx_j; тогда тензор Y получит приращение $\Delta Y = \{\Delta y_i\}$, состоящее из приращений

$\Delta y_i = y_i(\{x_j + \Delta x_j\}) - y_i(\{x_j\})$ его составляющих.

Линейная часть приращения Δy_i, т. е. **дифференциал**, определяется равенством $dy_i = \sum_j \frac{\partial y_i}{\partial x_j} \Delta x_j$. Отсюда следует, что тензор $dY = \{dy_i\}$, составленный из дифференциалов его составляющих, может быть записан в следующей форме:

$$dY = \left\{ \sum_j \frac{\partial y_i}{\partial x_j} \Delta x_j \right\} = \left\{ \frac{\partial y_i}{\partial x_j} \right\} \{\Delta x_j\} = \frac{dY}{dX} \Delta X.$$

Если X – зависимый тензор, то $dX = \frac{dX}{dX} \Delta X = E \Delta X = \Delta X$, т. е. $dX = \Delta X$, поэтому

$$dY = \frac{dY}{dX} dX. \qquad (П.10)$$

Формулы дифференцирования **сложных тензорных функций** аналогичны их скалярным вариантам:

$\frac{dY(X(Z))}{dZ} = \left(\frac{dY}{dX}\right)\left(\frac{dX}{dZ}\right)$ поэтому первый <u>дифференциал</u> имеет инвариантную форму (П.10) независимо от того, является X зависимым или независимым переменным.

Производные и дифференциалы высших порядков определяются как $\frac{d^k Y}{dX^k} = \frac{d}{dx}\left(\frac{d^{k-1}Y}{dX^{k-1}}\right), d^k Y = d(d^{k-1}Y)$.

Столь же легко обобщается дифференцирование скалярных функций k переменных на аналогичные тензорные:

$$\frac{\partial Y(X_1,\dots,X_n)}{\partial X_s} = \left\{\frac{\partial y_i(\{x_{j_1}\},\dots,\{x_{j_k}\})}{\partial x_{j_s}}\right\} = \left\{\frac{\partial y_i}{\partial x_{j_s}}\right\}. \quad (П.11)$$

При этом **полный дифференциал** определяется выражением

$$dY = \sum_{s=1}^{k} \frac{\partial Y}{\partial X_s} dX_s = \left\{\frac{\partial Y}{\partial X_s}\right\}\{dX_s\}. \quad (П.12)$$

Это определение можно распространить и на составные тензоры. Пусть $X = \{X_j\}$, где X_j – тензоры, тогда X называется **составным тензором**. Составной тензор, вообще говоря, не является тензором в обычном смысле. Действительно, пусть $X_1 = \{x_{1jk}\}$ – тензор размеров 2×3 и $X_2 = \{x_{2jk}\}$ – тензор размеров 4×5, тогда $X=\{X_1,X_2\}$ тензором в обычном смысле не является, так как эту совокупность чисел нельзя представить в виде $\{x_{ijk}\}$ потому, что в X есть, например, элемент x_{244}, но нет элемента x_{144}. В составном тензоре значения одних индексов ограничиваются

значениями других, а в несоставном каждый индекс принимает значения независимо от остальных. Совокупность $X = \{X_j\}$ является тензором, если все составляющие X_j – тензоры одинаковых размеров. Пусть $X = \{X_j\}$, $y = \{Y_i\}$ – составные тензоры и $Y = Y(X)$, тогда $\frac{\partial Y}{\partial X} = \left\{\frac{\partial Y_i}{\partial X_j}\right\}$, $dY = \left\{\frac{\partial Y_i}{\partial X_j}\right\}\{dX_j\} = \frac{dY}{dX}dX$, что является обобщением формул (П.11) и (П.12).

Отметим следующие правила тензорного дифференцирования:

$$\frac{d}{dX}B = C, \quad \frac{d}{dX}(BY) = B\frac{dY}{dX}, \quad \frac{d}{dX}(Y+Z) = \frac{dY}{dX}+\frac{dZ}{dX},$$

$$\frac{d}{dX}(YZ) = \frac{dY}{dX}Z + Y\frac{dZ}{dX},$$

$$\frac{d}{d\{x_j\}}(\{x_i\}\{a_{ij}\}\{x_j\}) = 2\{a_{ij}\}\{x_j\},$$

где B – произвольный постоянный тензор и $\{a_{ij}\} = \{a_{ji}\}$ – симметричный тензор.

Рассмотрим ряд Тейлора для скалярной функции двух скалярных переменных $y = y(x_1, x_2)$:

$$y(x_1+\Delta x_1, x_2+\Delta x_2)=y(x_1,x_2)+\frac{1}{1!}[y'_{x_1}(x_1,x_2)\Delta x_1+y'_{x_2}(x_1,x_2)\Delta x_2]+$$

$$+\frac{1}{2!}[y''_{x_1x_1}(x_1,x_2)\Delta x_1^2+y''_{x_1x_2}(x_1,x_2)\Delta x_1\Delta x_2+y''_{x_2x_1}(x_1,x_2)\Delta x_2\Delta x+y''_{x_2x_2}(x_1,x_2)\Delta x_2^2]+\ldots,$$

который можно представить в виде

$$y(X+\Delta X)=y(X)+\frac{1}{1!}\left\{\frac{\partial y}{\partial x_i}\right\}\{\Delta x_i\}+\frac{1}{2!}\left\{\frac{\partial^2 y}{\partial x_i\partial x_k}\right\}(\{\Delta x_i\}\{\Delta x_k\})+\ldots=$$

$$=y(X)+\frac{1}{1!}\frac{dy(X)}{dX}\Delta X+\frac{1}{2!}\frac{d^2y(X)}{dX^2}\Delta X^2+\ldots,$$

где тензор $X=\{x_i : i=1,2\}$; тензор приращений $\Delta X=\{\Delta x_i : i=1,2\}$ и ΔX^n – прямое n-кратное произведение $\Delta X\times\Delta X\times\ldots\times\Delta X$. Полученное выражение – полная аналогия ряда Тейлора скалярной функции одной скалярной переменной. В несколько более общем случае **скалярной функции любого тензорного аргумента** имеем разложение

$$y(X+\Delta X)=y(X)+\sum_{n=1}^{\infty}\frac{1}{n!}\frac{d^n y(X)}{dX^n}\Delta X^n$$

. (П.13)

Если задана система таких функций, составляющая тензорную функцию тензорного аргумента $Y(X)$, то, записывая (П.13) для каждой из них и образуя тензоры из однотипных членов этих рядов, получаем **тензорный ряд Тейлора**

$$Y(X+\Delta X)=Y(X)+\sum_{n=1}^{\infty}\frac{d^n Y(X)}{dX}\Delta X^n$$

, (П.14)

по форме не отличающийся от ряда Тейлора скалярной функции одного скалярного аргумента.

П.1.2. Функции от матриц

Постановка задачи

Пусть A = (a_{ij}) – квадратная матрица n-го порядка и f(l) – функция скалярного аргумента. Требуется определить, что следует понимать под f(A), т. е. требуется распространить функцию f(l) на матричные значения аргумента.

Эта задача очень просто решается, если $f(\lambda)=a_0\lambda^i+a_1\lambda^{i-1}+\ldots+a_i$ – многочлен, тогда, очевидно, можно положить $f(A)=a_0A^i+\ldots+a_iE$. Например, если $f(\lambda)=\lambda^2+3\lambda+4$ и $A=\begin{pmatrix}1&2\\3&4\end{pmatrix}$, то

$$f(A)=A^2+3A+4E=\begin{pmatrix}1&2\\3&4\end{pmatrix}^2+3\begin{pmatrix}1&2\\3&4\end{pmatrix}+4\begin{pmatrix}1&0\\0&1\end{pmatrix}=\begin{pmatrix}14&16\\24&38\end{pmatrix}$$

. Как же решить эту задачу в общем случае? Например, что такое $\sin\begin{pmatrix}1&2\\3&4\end{pmatrix}$ или $\ln\begin{pmatrix}1&2\\3&4\end{pmatrix}$?

Определение функций от матриц через многочлены

Определение 1: Скалярный многочлен $h(\lambda)$ называется **аннулирующим многочленом** квадратной матрицы A, если $h(A)=0$. Например, $h(\lambda)=\lambda^2-9\lambda+14$ – аннулирующий

многочлен матрицы $A=\begin{pmatrix}4 & 3\\ 2 & 5\end{pmatrix}$. Действительно, $A^2=\begin{pmatrix}22 & 27\\ 18 & 31\end{pmatrix}$,

$$h(A)=A^2-9A+14E=\begin{pmatrix}22 & 27\\ 18 & 31\end{pmatrix}-9\begin{pmatrix}4 & 3\\ 2 & 5\end{pmatrix}+14\begin{pmatrix}1 & 0\\ 0 & 1\end{pmatrix}=\begin{pmatrix}0 & 0\\ 0 & 0\end{pmatrix}=0$$

.

Если $h(\lambda)$ – аннулирующий многочлен матрицы A, то $g(\lambda)=h(\lambda)p(\lambda)$ есть аннулирующий многочлен матрицы A при любом многочлене $p(\lambda)$. Действительно,

$$g(A)=h(A)p(A)=0\cdot p(A)=0$$

.

Определение 2: Аннулирующий многочлен $\psi(\lambda)$ наименьшей степени с единичным старшим коэффициентом называется **минимальным многочленом** матрицы A.

Определение 3: **Характеристическим многочленом (определителем)** квадратной матрицы A называется

$$\Delta(\lambda)=|\lambda E-A|$$

.

Характеристическим (вековым) уравнением матрицы A называется уравнение $\Delta(\lambda)=0$. Корни характеристического уравнения называются **характеристическими (собственными) числами** матрицы A. Их совокупность называется **спектром матрицы**.

Теорема 1. (Гамильтона-Кэли). Всякая квадратная матрица удовлетворяет своему характеристическому уравнению, т. е.

$$\Delta(A)=0$$

.

Отсюда следует, что всякая квадратная матрица имеет аннулирующие многочлены (в частности, таковым является характеристический многочлен).

Теорема 2. Произвольный аннулирующий многочлен матрицы делится без остатка на ее минимальный многочлен.

Теорема 3. Минимальный многочлен существует и единственен для любой квадратной матрицы.

Теорема 4. Корнями минимального многочлена $\psi(\lambda)$ служат все характеристические числа матрицы A.

Таким образом, если

$$\Delta(\lambda)=(\lambda-\lambda_1)^{n_1}(\lambda-\lambda_2)^{n_2}\dots(\lambda-\lambda_k)^{n_k}$$

, где $\lambda_i\neq\lambda_j$ при $i\neq j$ и $n_1+\dots+n_k=n$, то минимальным многочленом является $\psi(\lambda)=(\lambda-\lambda_1)^{m_1}(\lambda-\lambda_2)^{m_2}\dots(\lambda-\lambda_k)^{m_k}$, где $0<m_i\le n_i$.

Теорема 5. Пусть $D(\lambda)$ – наибольший общий делитель всех миноров порядка n-1 характеристической матрицы $\lambda E-A$, тогда минимальный многочлен $\psi(\lambda)=\Delta(\lambda)/D(\lambda)$.

Пример 1. Найти минимальный многочлен $\psi(\lambda)$ матрицы

$$A=\begin{pmatrix}4 & 3\\ 2 & 5\end{pmatrix}$$

.

Решение. Характеристическая матрица:

$$\lambda E-A=\lambda\begin{pmatrix}1 & 0\\ 0 & 1\end{pmatrix}-\begin{pmatrix}4 & 3\\ 2 & 5\end{pmatrix}=\begin{pmatrix}\lambda-4 & -3\\ -2 & \lambda-5\end{pmatrix};$$

$$\Delta(\lambda)=\begin{vmatrix}\lambda-4 & -3\\ -2 & \lambda-5\end{vmatrix}=\lambda^2-9\lambda+14=(\lambda-2)(\lambda-7);$$

миноры порядка n-1=2-1=1: $M_{11}=\lambda-4$, $M_{12}=-2$, $M_{21}=3$, $M_{22}=\lambda-5$, их наибольший общий делитель $D(\lambda)=1$.

Отсюда

$$\varphi(\lambda)=\Delta(\lambda)/D(\lambda)=(\lambda^2-9\lambda+14)/1=(\lambda-2)(\lambda-7).$$

Пример 2. Найти минимальный многочлен матрицы

$$A=\begin{pmatrix}3 & -3 & 2\\ -1 & 5 & -2\\ -1 & 3 & 0\end{pmatrix}.$$

Первый способ.

$$\lambda E-A=\begin{pmatrix}\lambda-3 & 3 & -2\\ 1 & \lambda-5 & 2\\ 1 & -3 & \lambda\end{pmatrix};$$

$$\Delta(\lambda)=\lambda^3-8\lambda^2+20\lambda-16=(\lambda-2)^2(\lambda-4);$$

миноры порядка n-1=3-1=2:

$$M_{11}(\lambda)=\begin{vmatrix}\lambda-5 & 2\\ -3 & \lambda\end{vmatrix}=\lambda^2-5\lambda+6=(\lambda-2)(\lambda-3),$$

$$M_{12}(\lambda)=\begin{vmatrix}1 & 2\\ 1 & \lambda\end{vmatrix}=\lambda-2,$$

$$M_{13}(\lambda)=\begin{vmatrix}1 & \lambda-5\\ 1 & -3\end{vmatrix}=-\lambda+2=-(\lambda-2),$$

$$M_{21}(\lambda)=\begin{vmatrix}3 & -2\\ -3 & \lambda\end{vmatrix}=3\lambda-6=3(\lambda-2),$$

$$M_{22}(\lambda)=\begin{vmatrix}\lambda-3 & -2\\ 1 & \lambda\end{vmatrix}=\lambda^2-3\lambda+2=(\lambda-2)(\lambda-1),$$

$$M_{23}(\lambda)=\begin{vmatrix}\lambda-3 & 3\\ 1 & -3\end{vmatrix}=-3\lambda+6=-3(\lambda-2),$$

$$M_{31}(\lambda)=\begin{vmatrix}3 & -2\\ \lambda-5 & 2\end{vmatrix}=2\lambda-4=2(\lambda-2),$$

$$M_{32}(\lambda)=\begin{vmatrix}\lambda-3 & -2\\ 1 & 2\end{vmatrix}=2\lambda-4=2(\lambda-2),$$

$$M_{33}(\lambda)=\begin{vmatrix}\lambda-3 & 3\\ 1 & \lambda-5\end{vmatrix}=\lambda^2-8\lambda-12=(\lambda-2)(\lambda-6).$$

Их наибольший общий делитель: $D(\lambda)=\lambda-2$. Поэтому

$$\varphi(\lambda)=\frac{\Delta(\lambda)}{D(\lambda)}=\frac{(\lambda-2)^2(\lambda-4)}{\lambda-2}=(\lambda-2)(\lambda-4)=\lambda^2-6\lambda+8.$$

Второй способ. $\Delta(\lambda)=(\lambda-2)^2(\lambda-4)$. По теореме 4 минимальным многочленом может быть один из многочленов $\varphi_1(\lambda)=(\lambda-2)(\lambda-4)$ или $\varphi_2(\lambda)=(\lambda-2)^2(\lambda-4)$. Следует проверить, какие из этих двух многочленов являются аннулирующими (второй тривиально является), и выбрать из них многочлен минимальной степени. Таким образом, следует проверить, является ли $\varphi_1(\lambda)=(\lambda-2)(\lambda-4)=\lambda^2-6\lambda+8$ аннулирующим:

$$\psi_1(A)=(A-2)(A-4)=A^2-6A+8E=\begin{pmatrix}10&-18&12\\-6&22&-12\\-6&18&-8\end{pmatrix}-6\begin{pmatrix}3&-3&2\\-1&5&-2\\-1&3&0\end{pmatrix}+8\begin{pmatrix}1&0&0\\0&1&0\\0&0&1\end{pmatrix}=\begin{pmatrix}0&0&0\\0&0&0\\0&0&0\end{pmatrix}=0.$$

Ттаким образом, $\psi(\lambda)=\lambda^2-6\lambda+8$, как и в первом способе решения.

Пусть

$$\psi(\lambda)=(\lambda-\lambda_1)^{m_1}(\lambda-\lambda_2)^{m_2}\dots(\lambda-\lambda_k)^{m_k} \quad (\text{П.15})$$

минимальный многочлен матрицы А. Степень этого многочлена есть $m=m_1+\dots+m_k$. Пусть $g(\lambda)$ и $h(\lambda)$ – такие многочлены, что

$$g(A)=h(A). \quad (\text{П.16})$$

Тогда разность $\delta(\lambda)=g(\lambda)-h(\lambda)$ есть аннулирующий многочлен матрицы А, поэтому делится на $\psi(\lambda)$ без остатка, т. е.

$$g(\lambda)=h(\lambda) \pmod{\psi(\lambda)}. \quad (\text{П.17})$$

Отсюда и из (П.15) следует

$$\delta(\lambda_i)=0,\delta'(\lambda_i)=0,\dots,\delta^{(m_i-1)}(\lambda_i)=0,i=1,\dots,k.$$

Следовательно,

$$g(\lambda_i)=h(\lambda_i),\ g'(\lambda_i)=h'(\lambda_i),$$
$$g^{(m_i-1)}(\lambda_i)=h^{(m_i-1)}(\lambda_i),\ i=1,\dots,k. \quad (\text{П.18})$$

Пусть $f(\lambda)$ – некоторая функция, а

$\psi(\lambda)=(\lambda-\lambda_1)^{m_1}(\lambda-\lambda_2)^{m_2}\dots(\lambda-\lambda_k)^{m_k}$ –минимальный многочлен матрицы А. Совокупность из m чисел

$$f(\lambda_i),\ f'(\lambda_i),\ \dots,\ f^{(m_i-1)}(\lambda_i),\ i=1,\dots,k \quad (\text{П.19})$$

называется **значениями функции $f(\lambda)$ на спектре матрицы А** и обозначается $f(\Lambda_A)$. Если все значения (П.19) существуют, то говорят, что **функция $f(\lambda)$ определена на спектре матрицы А.**

Пример 3. Если $\psi(\lambda)=(\lambda-2)^2(\lambda+3)$, то для определенности $f(\lambda)$ на спектре должны существовать $f(1)$, $f'(1)$ и $f(-3)$. Например, фкнкция $f(\lambda)=\ln(\lambda)$ не определена, а функции $f(\lambda)=\ln(\lambda+4)$ и $f(\lambda)=\sin\lambda$ определены на спетре матрицы.

Равенства (П.18) означают, что многочлены $g(A)$ и $h(A)$ имеют одни и те же значения на спектре матрицы А, что будем записывать в виде $g(\Lambda_A)=h(\Lambda_A)$. И наоборот, из (П.18) следует (П.17) и (П.16).

Таким образом, если задана матрица А, то значения многочлена $g(\lambda)$ на ее спектре полностью определяют значение $g(A)$, т. е. все многочлены $g(\lambda)$, принимающие одни и те же значения на спектре матрицы А, имеют одно и то же значение $g(A)$. Поэтому естественно потребовать, чтобы определение $f(A)$ подчинялось этому принципу: значения $f(\lambda)$ на спектре матрицы А должны полностью определять значения $f(A)$, т. е. все функции $f(\lambda)$, совпадающие на спектре матрицы А, должны иметь одно и то же значение $f(A)$. Если

исходить из этого принципа, то в общем случае достаточно для определения $f(A)$ найти любой многочлен $g(\lambda)$, совпадающий с $f(\lambda)$ на спектре матрицы A, и положить $f(A) = g(A)$. Таким образом, приходим к следующему определению.

Определение 4. Если функция $f(\lambda)$ определена на спектре матрицы A и $g(\lambda)$ – любой многочлен, совпадающий с $f(\lambda)$ на спектре матрицы A (т. е. $f(\lambda_i) = g(\lambda_i)$), то, по определению,

$$f(A) = g(A). \qquad (П.20)$$

Такой многочлен можно получить, используя различные методы интерполяции или метод неопределенных коэффициентов. Среди таких многочленов существует единственный, имеющий степень, меньшую m. Таким образом, $f(A)$ выражается через E, A, A^2, …,A^{m-1} с некоторыми скалярными коэффициентами:

$$f(A) = a_0 A^{m-1} + a_1 A^{m-2} + \ldots + a_{m-1} E. \qquad (П.21)$$

Если использовать метод неопределенных коэффициентов, то, полагая $g(\lambda) = a_0 \lambda^{m-1} + a_1 \lambda^{m-2} + \ldots + a_{m-1}$, получим для нахождения коэффициентов $a_0,\ldots,a_{m-1}$ систему из m линейных уравнений:

$$g^{(j)}(\lambda_i) = f^{(j)}(\lambda_i), \quad i = 1,\ldots,k, \quad j = 0,1,\ldots,m_i - 1. \qquad (П.22)$$

Пример 4. Возьмем матрицу из примера 1: $A = \begin{pmatrix} 4 & 3 \\ 2 & 5 \end{pmatrix}$, ее минимальный многочлен есть $\psi(\lambda) = (\lambda - 2)(\lambda - 7)$.

Следовательно, для определения $f(A)$ достаточно найти любой многочлен $g(\lambda)$ такой, что $f(2) = g(2)$ и $f(7) = g(7)$.

а) Пусть $f(\lambda) = 8$. Тогда можно взять $g(\lambda) = 8$. Отсюда $f(A) = 8 \cdot E = \begin{pmatrix} 8 & 0 \\ 0 & 8 \end{pmatrix}$. И вообще, если $f(\lambda)$ – многочлен, то решение тривиально.

б) Пусть $f(\lambda) = 1/\lambda = \lambda^{-1}$, т. е. будем искать $f(A) = A^{-1}$ – обратную матрицу. Функция $f(\lambda)$ определена на спектре матрицы A, поэтому для определения $f(A)$ достаточно найти любой многочлен $g(\lambda)$ такой, что $f(2) = g(2) = 1/2$ и $f(7) = g(7) = 1/7$. Возьмем $g(\lambda) = a\lambda + b$, тогда $\begin{cases} 2a + b = 1/2 \\ 7a + b = 1/7 \end{cases}$, $a = -1/14$, $b = 9/14$, $g(\lambda) = (9 - \lambda)/14$. Следовательно,

$$f(A) = A^{-1} = \frac{1}{14}(9E - A) = \frac{1}{14}\left[9\begin{pmatrix} 1 & 0 \\ 0 & 1 \end{pmatrix} - \begin{pmatrix} 4 & 3 \\ 2 & 5 \end{pmatrix}\right] = \frac{1}{14}\begin{pmatrix} 5 & -3 \\ -2 & 4 \end{pmatrix}.$$

Проверка: $\frac{1}{14}\begin{pmatrix} 5 & -3 \\ -2 & 4 \end{pmatrix} \cdot \begin{pmatrix} 4 & 3 \\ 2 & 5 \end{pmatrix} = \begin{pmatrix} 1 & 0 \\ 0 & 1 \end{pmatrix}$.

Вместо многочлена $g(\lambda) = (9 - \lambda)/14$ можно найти другой подходящий многочлен. Возьмем, например, $h(\lambda) = a\lambda^2 + b$, тогда для нахождения его коэффициентоф имеем систему уравнений $\begin{cases} h(2) = 4a + b = 1/2 \\ h(7) = 49a + b = 1/7 \end{cases}$,

решение которой есть $a = -1/126, b = 67/126$.

Отсюда $h(\lambda) = (67 - \lambda^2)/126$ и

$$f(A) = A^{-1} = \frac{1}{126}(67E - A^2) = \frac{1}{126}\left[67\begin{pmatrix}1 & 0\\ 0 & 1\end{pmatrix} - \begin{pmatrix}22 & 27\\ 18 & 31\end{pmatrix}\right] = \frac{1}{126}\begin{pmatrix}45 & -27\\ -18 & 36\end{pmatrix} = \frac{1}{14}\begin{pmatrix}5 & -3\\ -2 & 4\end{pmatrix}.$$

результат тот же.

в) Вычислить $\sin(\pi A/2)$.

г) Вычислить eA и 2A.

д) Вычислить An.

е) Выразить $f(A)$ в общем случае через $f(2)$ и $f(1)$. Ответ для этого варианта: $f(A) = 0.2[(f(1) - f(2))A + (1f(2) - 2f(1))E]$.

Определение функций от матриц через компоненты матрицы

Недостатком определения $f(A)$, сделанного в предыдущем параграфе, является необходимость сложного вычисления коэффициентов в (П.21) для каждой функции $f(\lambda)$, что очень неудобно, если $f(A)$ нужно вычислить для множества различных функций $f(\lambda)$. Получим другое определение $f(A)$, свободное от этого недостатка.

Из линейности уравнений системы (П.22) следует, что коэффициенты $a_0, a_1, \ldots, a_{m-1}$ в (П.21) линейно зависят от значений $f^{(j)}(\lambda_i)$, т. е.

$$f(A) = \left(\sum_{ij} \alpha_{ij} f^{(j)}(\lambda_i)\right)A^{m-1} + \left(\sum_{ij} \alpha_{ij} f^{(j)}(\lambda_i)\right)A^{m-2} + \ldots + \left(\sum_{ij} \alpha_{i,m-1,j} f^{(j)}(\lambda_i)\right)E. \quad (П.23)$$

Группируя (П.23) относительно $f^{(j)}(\lambda_i)$, получим другую формулу:

$$f(A) = \sum_{i=1}^{k}\left[f(\lambda_i)z_{i1} + f'(\lambda_i)z_{i2} + \ldots + f^{(m_i-1)}(\lambda_i)z_{im_i}\right] = \sum_{ij} f^{(j)}(\lambda_i)Z_{ij}, \quad (П.24)$$

где m матриц Z_{ij} определяются заданием матрицы A и не зависят от выбора функции $f(\lambda)$.

Определение 4. Матрицы Z_{ij} из (П.24) называются **составляющими матрицами матрицы A или ее компонентами**. Формула (П.24) считается основной формулой для определения $f(A)$.

Теорема 6. Компоненты матрицы линейно независимы между собой и перестановочны между собой и с матрицей A.

Для нахождения компонент матрица A можно выполнить группировку (П.23)®(П.24) или же использовать (П.24) для нескольких функций $f(\lambda)$.

Пример 5. Рассмотрим снова матрицу A из примеров 1 и 4 и выразим $f(A)$ через компоненты матрицы A.

Первый способ. Представим $f(A)$ в виде (П.24), т. е.

$$f(A) = f(2)Z_1 + f(7)Z_2. \quad (П.25)$$

Используя решение примера 4е, имеем

$$f(A) = 0.2[(f(7) - f(2))A + (7f(2) - 2f(7))E] = f(2)[-0.2A + 1.4E] + f(7)[0.2A - 0.4E],$$

т. е.

$$Z_1 = -0.2A + 1.4E = 0.2\begin{pmatrix} 3 & -3 \\ -2 & 2 \end{pmatrix},\quad Z_2 = 0.2A - 0.4E = 0.2\begin{pmatrix} 2 & 3 \\ 2 & 3 \end{pmatrix}.$$

Таким образом, для любой функции $f(\lambda)$, определенной при l=2 и l=7, имеем

$$f\begin{pmatrix} 4 & 3 \\ 2 & 5 \end{pmatrix} = \frac{1}{5}\left[f(2)\begin{pmatrix} 3 & -3 \\ -2 & 2 \end{pmatrix} + f(7)\begin{pmatrix} 2 & 3 \\ 2 & 3 \end{pmatrix}\right].$$

Проверьте полученный результат для функций $f(\lambda) = 1$, $f(\lambda) = \lambda$, $f(\lambda) = \lambda^{-1}$.

Вычислите $\ln A$, An, (A-E)n.

Второй способ. Положим в (1.11) $f(\lambda) = \lambda - 2$, тогда $f(2) = 2 - 2 = 0$, $f(7) = 7 - 2 = 5$ и $f(A) = A - 2E$, отсюда $A - 2E = 5Z_2$. Аналогично, при $f(\lambda) = \lambda - 7$ имеем $f(2) = 2 - 7 = -5$, $f(7) = 7 - 7 = 0$, $f(A) = A - 7E = -5Z_1$.

Таким образом, имеем систему уравнений

$$\begin{cases} -5Z_1 = A - 7E \\ 5Z_2 = A - 2E \end{cases},$$

откуда $Z_1 = -0.2A + 1.4E,\ Z_2 = 0.2A - 0.4E$, что совпадает с решением первым способом.

Пример 6. Выразить $f(A)$ через компоненты матрицы

$$A = \begin{pmatrix} 2 & -1 & 1 \\ 0 & 1 & 1 \\ -1 & 1 & 1 \end{pmatrix}.$$

Представление функций от матриц рядами

В теории степенных рядов рассматривается представление скалярных функций в виде

$$f(\lambda) = \sum_{k=0}^{\infty} \alpha_k (\lambda - \lambda_0)^k.$$

(П.26)

Все члены ряда в правой части (П.26) будут определены, если скалярный аргумент заменить на матричный. Поэтому представляется естественным определить функцию от матрицы с помощью степенного ряда, т. е. положить

$$f(A) = \sum_{k=0}^{\infty} \alpha_k (A - \lambda_0 E)^k.$$

(П.27)

Однако при этом возникают вопросы сходимости ряда в (П.27) к $f(A)$, т. е. частичные суммы должны иметь своим пределом $f(A)$:

$$\lim_{p \to \infty} \sum_{k=0}^{p} \alpha_k (A - \lambda_0 E)^k = f(A).$$

Теорема 7. Если функция $f(\lambda)$ разлагается в степенной ряд (П.26) в круге $|\lambda - \lambda_0| < r$ на комплексной плоскости, то это разложение выполняется (имеет место (П.27)) и для любой матрицы A, все

характеристические числа которой лежат внутри этого круга сходимости, т. е. если $|\lambda_i - \lambda_0| < r$, $i = \overline{1,n}$.

Из этой теоремы и известных разложений следуют формулы:

$$e^A = \sum_{k=0}^{\infty} \frac{1}{k!} A^k, \qquad \sin A = \sum_{k=0}^{\infty} (-1)^k \frac{A^{2k+1}}{(2k+1)!},$$

$$\cos A = \sum_{k=0}^{\infty} (-1)^k \frac{A^{2k}}{(2k)!},$$

$$(E - A)^{-1} = \sum_{k=0}^{\infty} A^k \quad (|\lambda_i| < 1, i = \overline{1,n}),$$

$$\ln A = \sum_{k=1}^{\infty} \frac{(-1)^{k-1}}{k} (A - E)^k \quad (|\lambda_i| < 1, i = \overline{1,n}).$$

Интегральное представление функций от матриц

В теории функций комплексного переменного известна интегральная формула Коши

$$f(\lambda) = \frac{1}{2\pi i} \oint_{\Gamma} \frac{f(z)}{z - \lambda} dz,$$

где $f(\lambda)$ – аналитическая функция внутри контура Г; z – комплексный аргумент; l – точка внутри контура Г.

Оказывается, что эту формулу можно распространить и на матричные аргументы:

$$f(A) = \frac{1}{2\pi i} \int_{\Gamma} (zE - A)^{-1} f(z) dz \qquad (П.28)$$

при условии, что характеристические числа матрицы A находятся внутри Г.

Некоторые свойства функций от матриц

1. Все приведенные выше определения функций от матриц эквивалентны в том смысле, что они определяют одно и то же значение $f(A)$.

2. Для диагональных матриц имеем

$$f\begin{pmatrix} \lambda_1 & & 0 \\ & \ddots & \\ 0 & & \lambda_n \end{pmatrix} = \begin{pmatrix} f(\lambda_1) & & 0 \\ & \ddots & \\ 0 & & f(\lambda_n) \end{pmatrix}.$$

3. Пусть $g(\lambda) = G(f_1(\lambda), \ldots, f_k(\lambda))$, где функции $f_1, \ldots, f_k$ определены на спектре матрицы A, а функция $G(u_1, \ldots, u_k)$ может содержать действия сложения, умножения, умножения на число и замены величины на произвольную функцию от нее (суперпозиция). Тогда из $g(A_A) = 0$ следует $G(f_1(A), \ldots, f_k(A)) = 0$.

Например, пусть A – неособенная матрица ($|A| \neq 0$). Тогда в скалярном тождестве $e^{\ln \lambda} - \lambda = 0$ можно заменить l на A: $e^{\ln A} - A = 0$.

Свойство 3 не очень строго можно сформулировать следующим образом. При выполнении не очень сильных ограничений соотношения между функциями скалярного аргумента сохраняют силу при переходе

к матричному аргументу. Например: $\sin^2 A + \cos^2 A = E$,
$e^A \cdot e^B = e^{A+B}$ и т.д.

Пример 7. Рассмотрим дифференциальное уравнение

$$\frac{dx}{dt} = ax \qquad (П.29)$$

с начальным условием $x(t_0) = x_0$. Его решение имеет вид

$$x(t) = e^{a(t-t_0)}x_0. \qquad (П.30)$$

Рассмотрим теперь систему уравнений

$$\begin{cases} \frac{dx_1}{dt} = a_{11}x_1 + a_{12}x_2 + \ldots + a_{1n}x_n, \\ \ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots \\ \frac{dx_n}{dt} = a_{n1}x_1 + a_{n2}x_2 + \ldots + a_{nn}x_n \end{cases} \qquad (П.31)$$

с начальными условиями $x_1(t_0) = x_{10}, \ \ldots, x_n(t_0) = x_{n0}$, что можно записать в матричном виде как

$$\frac{dX}{dt} = AX \qquad (П.32)$$

с начальным условием $X(t_0) = X_0$, где $X = \begin{pmatrix} x_1 \\ \vdots \\ x_n \end{pmatrix}$, $X_0 = \begin{pmatrix} x_{10} \\ \vdots \\ x_{n0} \end{pmatrix}$,

$A = \begin{pmatrix} a_{11} \ldots a_{1n} \\ \vdots \\ a_{n1} \ldots a_{nn} \end{pmatrix}$.

Уравнение (П.32) получается из (П.29) заменой скалярного аргумента на матричный. Применяя функции от матриц, можно показать, что и решение уравнения (П.32) имеет вид, аналогичный решению (П.30):
$X(t) = e^{A(t-t_0)}X_0$.

Функции от матриц вида H(a,b)

Рассмотрим квадратные матрицы m-го порядка вида

$$H(a,b) = aE + bI = \begin{pmatrix} a+b & b & \ldots & b \\ b & a+b & \ldots & b \\ \cdot & \cdot & \ldots & \cdot \\ b & b & \ldots & a+b \end{pmatrix}, \qquad (П.33)$$

где a и b – действительные числа; E – единичная матрица; I – матрица, состоящая из единиц. Определим функции от матриц этого вида, предполагая $b \neq 0$. Если $b = 0$, то это случай неинтересный.

Найдем сначала характеристический определитель матрицы H(a,b):

$$\Delta_n(\lambda) = |\lambda E - H(a,b)| = \begin{vmatrix} \lambda - a - b & -b & \ldots & -b \\ -b & \lambda - a - b & \ldots & -b \\ \cdot & \cdot & \ldots & \cdot \\ -b & -b & \ldots & \lambda - a - b \end{vmatrix}.$$

Сумма всех строк есть строка (l-a-mb, l-a-mb, …, l-a-mb), которая превращается в нулевую строку при l=a+mb. Следовательно, l1=a+mb – корень характеристического определителя. Далее, при l=a характеристический определитель превращается в определитель из одинаковых строк (-b, -b, …, -b), он равен нулю, поэтому l=a – тоже корень. Найдем производные от $\Delta_m(\lambda)$. Производная от определителя есть сумма определителей, в каждом из которых взята производная всех элементов одной из строк:

$$\Delta'_m(\lambda)=\begin{vmatrix} 1 & 0 & 0 & \dots & 0 \\ -b & \lambda-a-b & -b & \dots & 0 \\ \dots & \dots & \dots & \dots & \dots \\ -b & -b & -b & \dots & \lambda-a-b \end{vmatrix}+\begin{vmatrix} \lambda-a-b & -b & -b & \dots & 0 \\ 0 & 1 & 0 & \dots & 0 \\ -b & -b & \lambda-a-b & \dots & 0 \\ -b & -b & -b & \dots & \lambda-a-b \end{vmatrix}+\dots=m\Delta_{m-1}(\lambda).$$

т. е. $\Delta'_m(\lambda)=m\Delta_{m-1}(\lambda)$ и $\Delta'_m(a)=0$. Вторая производная:

$\Delta''_m(\lambda)=m\Delta'_{m-1}(\lambda)=m(m-1)\Delta_{m-2}(\lambda)$ и $\Delta''_m(a)=0$ и так далее. Наконец, последние производные:

$\Delta_m^{(m-2)}(\lambda)=m(m\quad 1)\dots 3\Delta_2(\lambda)$ и $\Delta_m^{(m-2)}(a)-0$;

$\Delta_m^{(m-1)}(\lambda)=m(m-1)..2\Delta_1(\lambda)=m(m-1)..2(\lambda-a-b)$ и

$\Delta_m^{(m-1)}(a)=m!b$; $\Delta_m^{(m)}(\lambda)=m!$. Таким образом,

$$\Delta_m(a)=\Delta'_m(a)=\Delta''_m(a)=\dots=\Delta_m^{(m-2)}(a)=0,$$

следовательно, l=a является корнем кратности m-1. Итак,

$\Delta_m(\lambda)=(\lambda-a-mb)(\lambda-a)^{m-1}$, т. е.

характеристические числа матрицы H(a,b) есть l1=a+mb, l2=a, …, lm=a.

Найдем теперь минимальный многочлен матрицы H(a,b). По теореме 4 он имеет вид $\psi(\lambda)=(\lambda-a-mb)(\lambda-a)^k$, где $1\le k\le m-1$. Испытаем простейший из возможных варианов – $\psi(\lambda)=(\lambda-a-mb)(\lambda-a)$, учитывая, что $I^2=mI$:

$$\psi(H(a,b))=[aE+bI-(a+mb)E][aE+bI-aE]=[bI-mbE]bI=b^2(I^2-mI)=0$$

.

Таким образом, $\psi(\lambda)=(\lambda-a-mb)(\lambda-a)$ – минимальный многочлен, поэтому

$$f(H(a,b))=f(\lambda_1)H_1+f(\lambda_2)H_2=f(a+mb)H_1+f(a)H_2,\qquad (П.33)$$

где H1 и H2 – компоненты матрицы H(a,b). Найдем эти компоненты, взяв функции $f(\lambda)=\lambda-\lambda_1$ и $f(\lambda)=\lambda-\lambda_2$:

$$\begin{Bmatrix} f(\lambda)=\lambda-\lambda_1 \\ f(\lambda)=\lambda-\lambda_2 \end{Bmatrix}:\begin{Bmatrix} H(a,b)-\lambda_1E=0H_1+(\lambda_2-\lambda_1)H_2 \\ H(a,b)-\lambda_2E=(\lambda_1-\lambda_2)H_1+0H_2 \end{Bmatrix}:\begin{Bmatrix} aE+bI-(a+bm)E=(a-a-bm)H_2 \\ aE+bI-aE=(a+bm-a)H_1 \end{Bmatrix}$$

, $bI=bmH_1, bI-bmE=-bmH_2$. Отсюда

$$H_1=\frac{1}{m}I,\qquad H_2=E-\frac{1}{m}I,\qquad (П.35)$$

$$f(H(a,b))=f(a+bm)\frac{1}{m}I+f(a)(E-\frac{1}{m}I)=f(a)E+\frac{f(a+bm)-f(a)}{m}I=$$
$$=H(f(a),\frac{f(a+bm)-f(a)}{m}).$$

(П.36)

Итак, любая функция от матрицы вида H(a,b) есть матрица того же вида.

Отметим следующие свойства компонент H1 и H2:

1) независимость от a и b;

2) симметричность: $H_1^T = H_1, H_2^T = H_2$; (П.37)

3) коммутативность: $H_1 H_2 = H_2 H_1$; (П.38)

4) идемпотентность: $H_1^2 = H_1, H_2^2 = H_2$,

$H_1^n = H_1, H_2^n = H_2$; (П.39)

5) взаимоаннулируемость (ортогональность):

$H_1 H_2 = H_2 H_1 = 0$. (П.40)

Из (П.36), в частности, получаем:

$$H^n(a,b) = (a+mb)^n H_1 + a^n H_2 = H\left(a^n, \frac{(a+mb)^n - a^n}{m}\right)$$

,

$$H^{-1}(a,b) = \frac{1}{a+mb} H_1 + \frac{1}{a} H_2 = H\left(\frac{1}{a}, \frac{-b}{a(a+mb)}\right)$$

, (П.41)

$$\sqrt{H(a,b)} = \sqrt{a+mb} H_1 + \sqrt{a} H_2 = H\left(\sqrt{a}, \frac{\sqrt{a+mb} - \sqrt{a}}{m}\right)$$

Из (П.31) – (П.38) следует:

$$H(a_1,b_1) + H(a_2,b_2) = H(a_1 + a_2, b_1 + b_2)$$

, (П.42)

$$H(a_1,b_1) H(a_2,b_2) = (a_1 + mb_1)(a_2 + mb_2) H_1 + a_1 a_2 H_2$$

. (П.43)

Таким образом, различные функции и операции от матриц вида H(a,b) легко вычисляются, что делает их удобными для различных применений.

ПРИЛОЖЕНИЕ 2

2.1. Растровые и векторные изображения

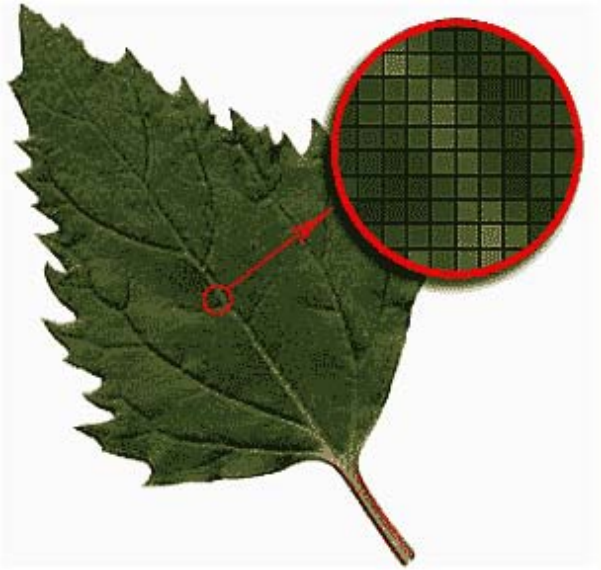
Компьютерная технология имеет свою специфику, которая заключается в том, что изображение должно быть преобразовано в цифровую форму или закодировано. В свою очередь, цифровая компьютерная графика по способу кодирования подразделяется на растровую (точечную) и векторную. Каждая из них имеет свои достоинства и недостатки.

Принцип точечной или растровой графики чрезвычайно прост. Он был изобретен и использовался людьми за много веков до появления компьютеров. Это - и рисование «по клеточкам» - способ переноса изображения с подготовленного картона на стену предназначенную для фрески, и такие направления искусства, как мозаика, витражи, вышивка. В любой из этих техник изображение строится из дискретных элементов.

В растровой графике, как и на экране телевизора или монитора, любое изображение состоит из совокупности очень мелких элементов (точек), которые называются пикселам (pixel). Слово «пиксел» - это аббревиатура от английских слов picture element (элемент изображения). Цвет каждого пикселя записывается в память компьютера при помощи определенного количества битов. Бит - минимальная единица памяти компьютера, которая может хранить либо значение 0, либо 1. Пиксель представляет собой наименьший адресуемый элемент растрового изображения.

Если говорят, что картинка имеет размер 800x600, то эти числа отражают количество пикселей по горизонтали (800) и вертикали (600). Чем больше количество пикселей в изображении, тем больше у него размер. В том числе и размер места, занимаемого на диске.

Чтобы хранить информацию о пикселе, на диске компьютера выделяется некоторое количество бит. Число цветов, в которые можно раскрасить отдельный пиксель, определяется двумя в степени, равной количеству битов, хранящих цветовую информацию о пикселе. В контрастной черно-белой картинке каждый пиксель кодируется одним битом, то есть пиксель может быть либо черным, либо белым. Восьмибитное изображение позволяет иметь 256 цветов, а 24 бита обеспечивают присутствие в изображении более 16 миллионов цветов, что дает возможность работать с изображениями профессионального качества.



Например, это изображение древесного листа описывается конкретным расположением и цветом каждой точки сетки, что создает изображение примерно так же, как в мозаике. Это значит, что, работая с растровыми изображениями, мы работаем над составляющими их группами пикселей. Растровые изображения обеспечивают высокую точность передачи градаций цветов и полутонов, а также высокую детализацию изображения, поэтому они являются оптимальным средством представления тоновых изображений, таких как фотографии.

Подобную мозаику вы можете увидеть и на любой фотографии, сильно её увеличив. В редакторе Photoshop это можно сделать, зажав клавишу "Alt" и покрутив при этом колесико мышки.

Достоинства растрового формата

Основным достоинством растровой графики является простота и, как следствие, техническая реализуемость автоматизации ввода или оцифровки изобразительной информации. Существует большое количество внешних устройств для ввода фотографий, слайдов, рисунков и т.д. - это сканеры, видеокамеры, цифровые фотоаппараты. Не менее важным достоинством точечной графики является фотореалистичность.

Недостатки растрового формата

Точечной графике присущи и некоторые недостатки. Если отсканировать не очень большую фотографию с максимальным разрешением и глубиной цвета одна картинка потребует для сохранения достаточно большой объем дискового пространства в несколько десятков мегабайтов. Объем файла точечной графики - это произведение площади изображения на разрешение и на глубину цвета.

Для иллюстрации представлены два изображения. Первое изображение имеет размер 550*359 и занимает объем 304 Кбайт:

550*359, объем 304 Кб.

Это же изображение уменьшенное по размеру (110*72) занимает на жестком диске значительно меньший объем (2,65 Кбайт):

110*72, объем 2,65 Кб.

Из этого примера видно, что растровое изображение высокого качества и большого размера может занимать значительный объем дискового пространства, что не приемлемо для изображений размещаемых в Интернете. Перед размещения в Интернете необходимо оптимизировать графические файлы, что всегда является компромиссом между качеством и объемом графического файла.

Заметим, что при сохранении изображений для печати и для Интернета используются достаточно противоположные принципы, а именно: фотография для печати должна нести в себе максимум информации - количеством цветов, количеством пикселей, чтобы при печати вся эта информация была отражена наиболее полно. Для передачи

информации на экран пользователя по сети Интернет - изображение должно быть максимально маленьким, опять же в смысле количества цветов и пикселей. Чем меньше изображение, тем быстрее оно дойдет до пользователя и тем меньше пользователь заплатит за трафик. По этой причине часто изображения, размещенные в сети Интернет не подходят для печати в газете или журнале.

Еще один недостаток проявится, когда потребуется слегка повернуть изображение с четкими тонкими вертикальными линиями. Обнаружится, что они превратились в ступеньки. Если же попытаться увеличить масштаб изображения, то можно увидеть, что оно станет не резким и на нем появятся пикселы. Это означает, что в растровой графике при любых трансформациях - поворотах, масштабировании, наклонах - нельзя обойтись без искажений.

Из программ растровой графики наиболее популярны **Adobe Photoshop, Corel Photo Paint.**

Векторная графика

Принцип кодирования графической информации в векторной графике принципиально отличается от растровой. В векторной графике все изображения описываются в виде математических объектов - контуров. Каждый контур представляет собой независимый объект, который можно перемещать, масштабировать, изменять множество раз. Все линии определяются начальными точками и формулами, описывающими сами линии. Поэтому при изменении размера рисунка пропорции и очертания всегда точно выдерживаются. В векторную графике изображение состоит из отдельных объектов - прямых и кривых линий, замкнутых и разомкнутых фигур, прямоугольников, эллипсов и т.п., каждый из которых имеет свои характеристики цвета, толщины контура, стиля линии и т.д.

Достоинства векторной графики

К достоинствам векторной графики, относится то, что она экономна в плане объемов дискового пространства, необходимого для хранения изображений. Это связано с тем, что сохраняется не само изображение, а только некоторые основные данные, используя которые, программа всякий раз воссоздает изображение заново. Кроме того, описание цветовых характеристик почти не увеличивает размера файла. Объекты

векторной графики легко трансформируются, причем практически без ущерба для качества изображения.

При увеличении размеров изображения не возникает искажений, пикселизации и объем занимаемый изображением не зависит от его размеров. В тех областях графики, где принципиальное значение имеет сохранение ясных и четких контуров, например, в шрифтовых композициях, в создании логотипов и др., векторные программы незаменимы. Векторная графика использует все преимущества разрешающей способности любого выводного устройства, например, принтера. Изображение всегда будет выглядеть настолько качественно, насколько способно данное устройство.

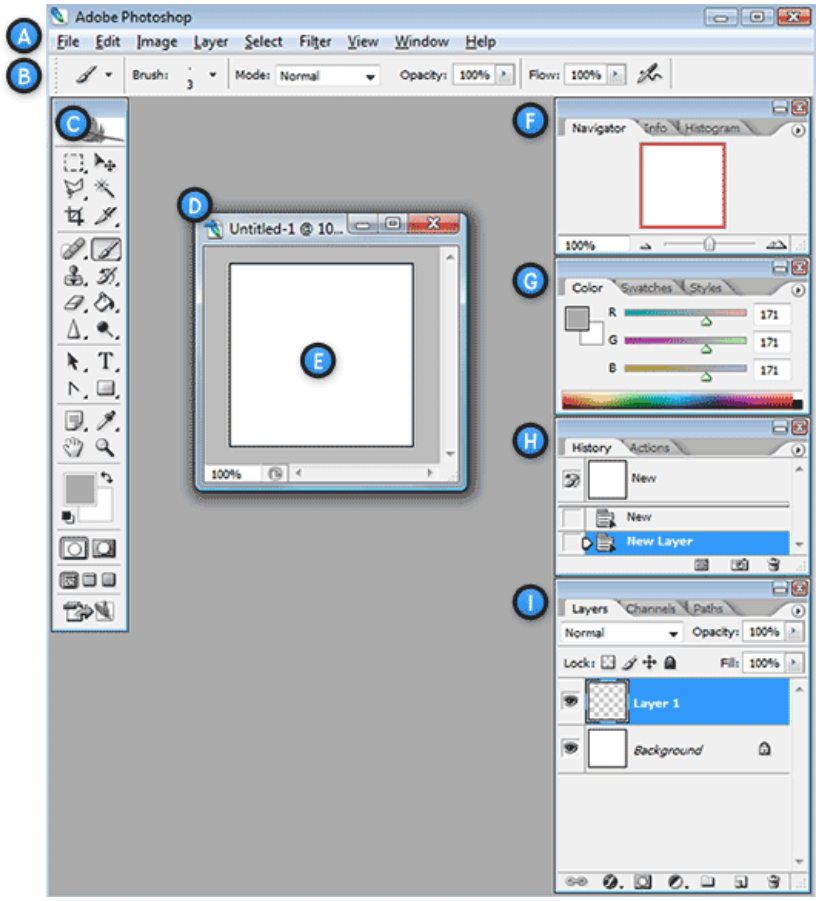
Недостатки векторной графики

Однако, с другой стороны, векторная графика ограничена в чисто живописных средствах: в таких программах практически невозможно создавать фотореалистические изображения.

Наиболее популярными программами векторной графики являются **CorelDRAW, Adobe Illustrator.**

Рабочая область в Photoshop

На рисунке ниже вы можете увидеть рабочую область графического редактора.



A. Горизонтальное меню - Организованная область меню, разделенная по различным выполняемым типам задач.

B. Панель настроек - Содержит настройки, возможные для выбранного инструмента.

C. Панель инструментов - Содержит широкий выбор доступных инструментов в Photoshop, а также цвет переднего плана, фоновый цвет и другие функции.

D. Окно документа - Окно, которое вмещает редактируемый в Photoshop документ.

E. Активная область изображения - Окно документа содержит активную область изображения (показана белым сверху). Это область, в которой работает художник.

F. Навигатор - Уменьшенная версия активной области изображения текущего документа. Используется для навигации по большим изображениям или при большом зуме (увеличении).

G. Панель цветов - Панель, используемая для выбора цветов переднего и заднего планов для рисования или заливки ими.

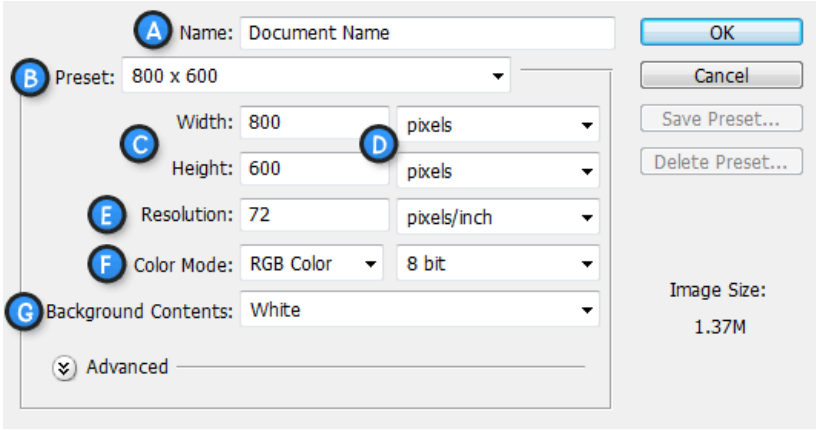
H. История - Постоянно обновляющаяся запись предыдущих изменений внесенных в документ. Используется для отмены сделанных шагов *(Undo)*

I. Слои - Слои в Photoshop позволяют вам работать над отдельными полотнами, позволяя разным вещам проявляться над или под другими.

Создание и сохранение нового документа

Чтобы создать новый документ в Photoshop, нажмите *File* > *New* из **Горизонтального меню** .

Появится диалоговое окно, такое, как показано на рисунке снизу:



A. Имя документа - Здесь укажите имя документа (по выбору)

B. Предустановки -Выберите размер документа из ранее установленных

C. Размеры документа - Здесь укажите ширину и высоту документа

D. Единицы измерения документа - Укажите единицы измерения документа. Для любой типичной работы для веба или на мониторе выбирайте пиксели.

E. Разрешение - Укажите разрешение документа. Разрешение обычно означает число точек (или пикселей) на дюйм. При печати вам лучше поставить значение повыше (300 или выше), что также увеличит и размер документа. Тем не менее, для экранной работы или работы в вебе, подходящее разрешение 72.

F. Цветовой режим - Укажите цветовой режим документа. Для печати используется режим CMYK , файлы для работы в интернете сохраняются в RGB . Пока вам лучше всего работать в режиме RGB Color (Red/Green/Blue).

G. Содержание Фона - Укажите цвет фона вашего документа.

Диалоговое окно, показанное выше, создаст документ размером 800x600 пикселей с белым фоном. Имя документа будет "Document Name", а разрешение будет наиболее подходящим для экранной работы.

Чтобы сохранить документ Photoshop, нажмите *File > Save* из **Горизонтального меню** . Во время сохранения, убедитесь, что ввели имя документа, которое позже узнаете и также выберите формат "Photoshop (*.PSD)" . Выбор этого формата обеспечит сохранение у документа всех атрибутов Photoshop, а также информацию о слоях, так чтобы вы смогли потом работать с ним.

Панель инструментов

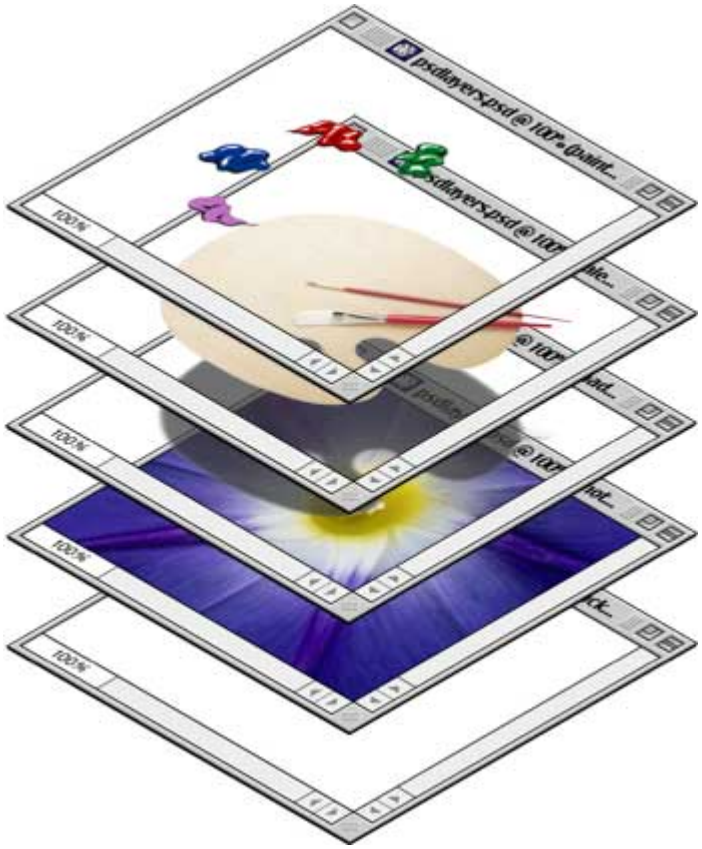
1) Move Tool – предназначен для выбора слоёв и манипуляциям с ними, например, с помощью него мы перетаскиваем слои по документу.
2) Rectangular Marquee Tool – предназначен для выделения области документа (у этого инструмента есть несколько разновидностей выделения: прямоугольником, овалом, колонкой или строкой).
3) Lasso Tool – он в свою очередь нужен для выделения более сложной формы. У этого инструмента, также есть несколько разновидностей: лассо – выделение будет следовать за траекторией курсора; полигональное лассо – выделение осуществляется с помощью указания вершин; магнитное лассо – оно само будет пытаться определить границу объекта, которое выделяется.
4) Magic Wand Tool – эта магическая палочка выделяет область одного цвета (чувствительность может задаваться).
5) Crop Tool – кадрирование области документа.
6) Pencil Tool – карандаш, название говорит само за себя, для рисования. Также есть другие разновидности: карандаш и кисть.
7) Clone Stamp Tool – клонирование области. Очень полезный инструмент для ретуширования фотографии.
8) History Brush Tool – оригинальный инструмент для взаимодействия с историей изменения документа. Проявляет в определённой области то, что было раньше. Например, до наложения фильтра резкости.
9) Eraser Tool – стирательная резинка
10) Gradient Tool – градиентная заливка. Вторая разновидность инструмента – заливка однородным цветом (инструмент применяется к выделенной области)
11) Blur Tool – размытие кистью. Другие разновидности этого инструмента: наведение резкости и вытягивание кистью.
12) Dodge Tool – высветление области. Другая разновидность инструмента: затемнение.
13) Pen Tool - перо. Предназначено для построения и редактирования пути.
14) Horizontal Type Tool - инструмент для создания/редактирования текста. Набирать текст можно, как горизонтальный, так и вертикальный.
15) Path ion Tool – тоже самое что и Move Tool, только предназначен для путей. Есть разновидность инструмента для выделения отдельных точек пути.
16) Ellipse Tool – создание геометрических фигур.
17) Notes Tool – создание заметок в документе.
18) Eyeper Tool – измерение цвета точки.
19) Hand Tool – «лапка» для перемещения по документу.

20) Zoom Tool – инструмент масштабирования документа (приближает/удаляет зрителя).
21) Default Foreground and Background Colors – устанавливает изначальные цвета для фона и переднего планов. Switch Foreground and Background Colors – меняет местами эти цвета между собой.
22) Set Foreground Color / Set Background Color – установленные значения цвета на данный момент.
23) Edit in Quick Mask Mode – включает/выключает режим быстрой маски.

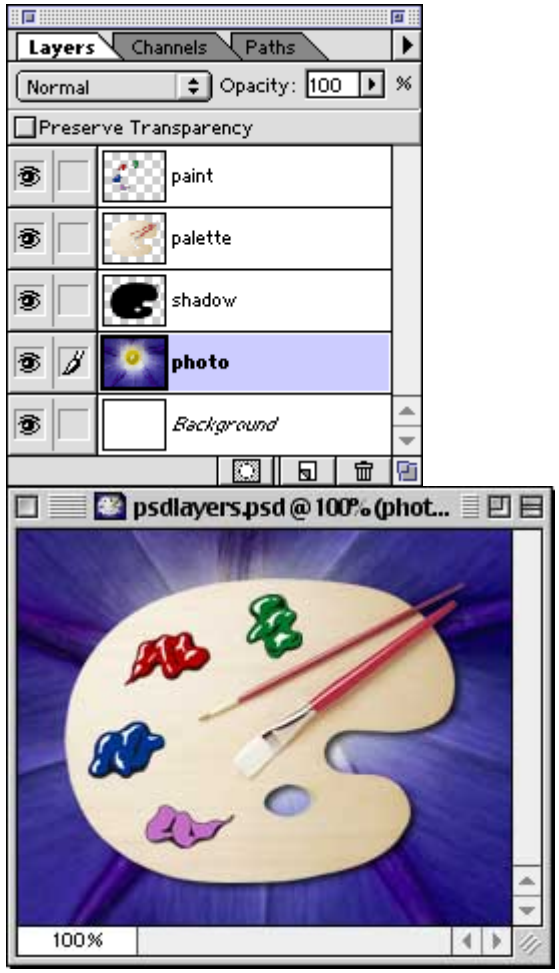
Заметьте, что за одним инструментом на панели «прячутся» еще несколько. Они становятся доступны, если нажать на маленький треугольник снизу.

Слои документа в Photoshop

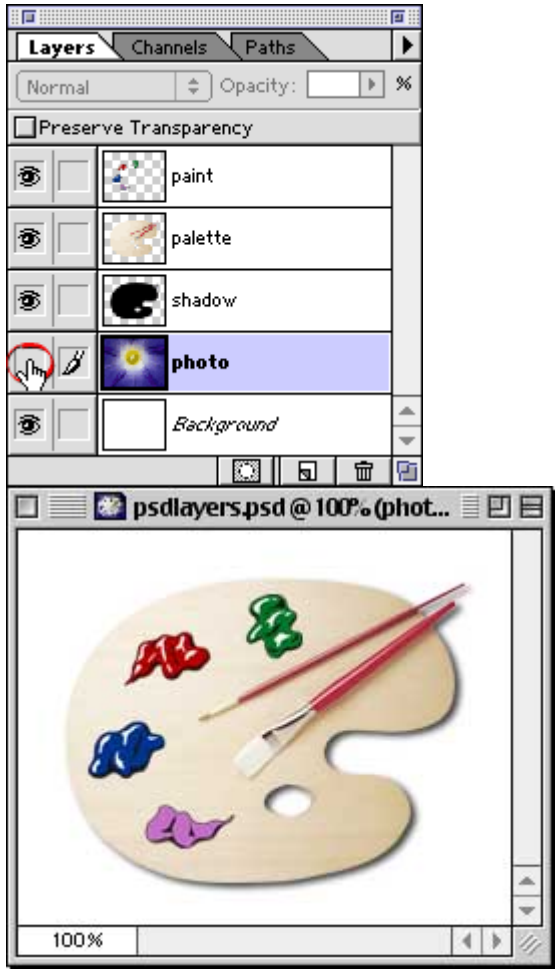
На рисунке снизу показана наглядно одна из главных концепций документа в Photoshop. Изображение представляет собой ничто иное, как слои изображений, наложенных друг на друга.



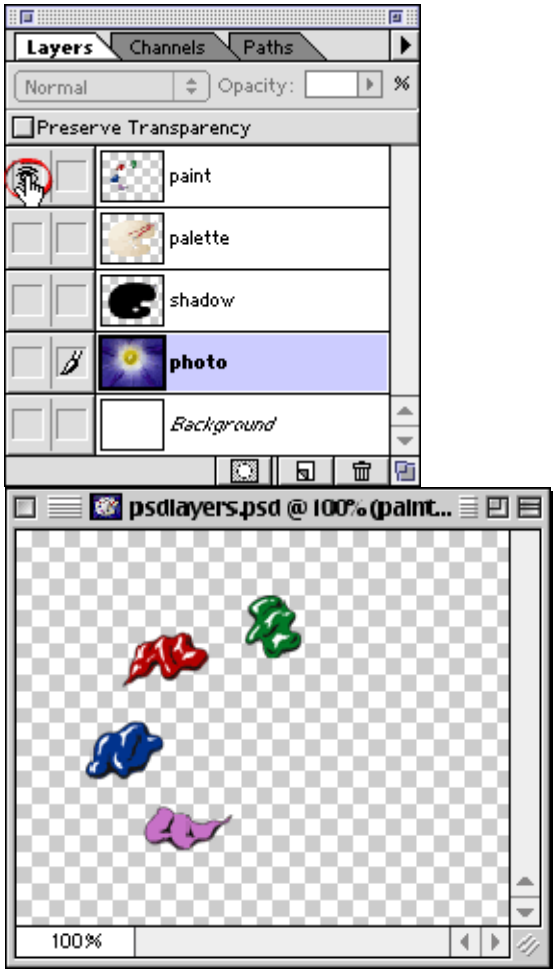
1. Слои управляются в палитре слоев **(Layers)** . В ней отображается небольшой эскиз (превью каждого слоя, чтобы было проще его найти). Отображающийся в Photoshop документ — это соединение всех слоев из стопки сверху вниз (внизу справа).



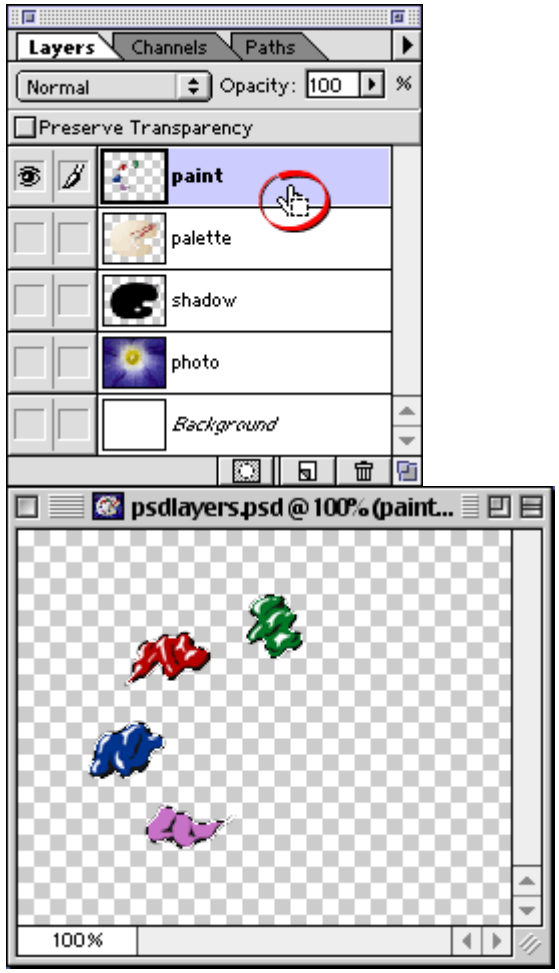
2. Вы можете включать и выключать отображение любого слоя. Значок в виде глаза слева от каждого слоя контролирует его видимость. Слой делается видимым или скрытым при нажатии этого значка (внизу слева). В результате тот же документ, что показан ранее, за исключением, одного слоя.



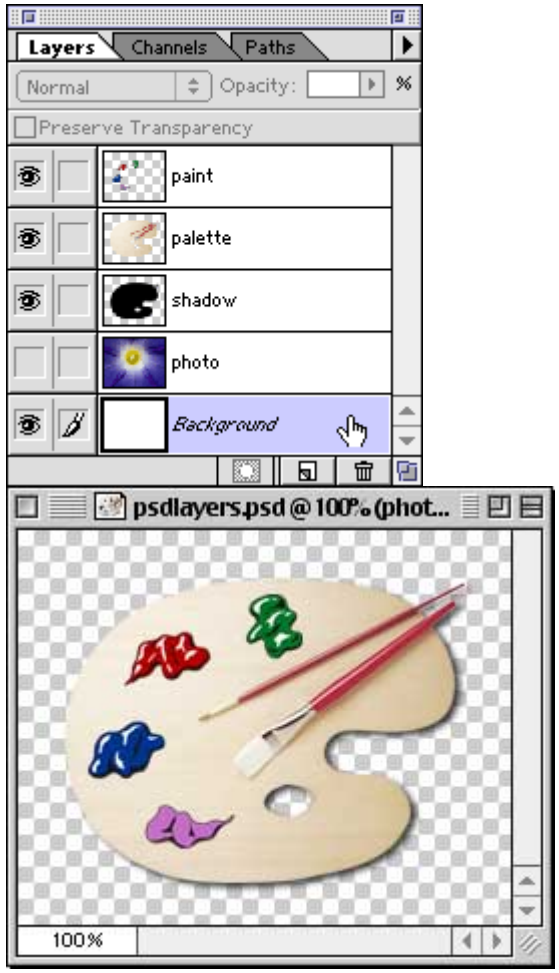
3. Подсвеченный слой в палитре Layers означает, что он является активным (или выбранным). Это означает, что все изменения в окне документа будут применяться к нему.



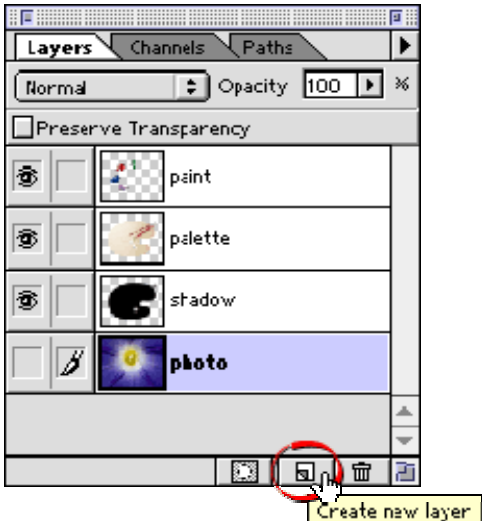
4. Вы можете быстро выделить все непрозрачные области слоя зажав *Ctrl* и кликнув по нужному слою **в палитре Layers.** Когда вы удерживайте клавишу *Ctrl* , указатель мыши превращается в руку с квадратом выделения (ниже слева). В результате изображение слоя выделится. Вы увидите «муравьиную дорожку» вокруг изображения слоя (внизу справа).

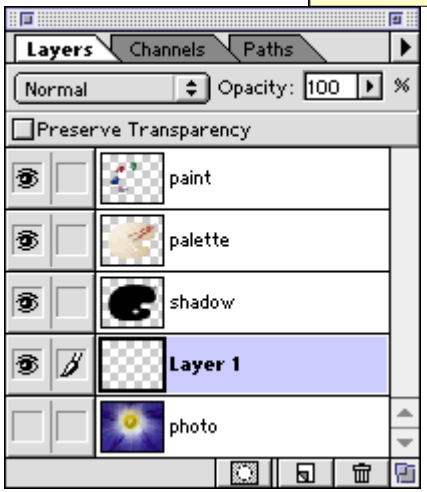


5. Чтобы удалить слой, щелкните по нему и перетащите в корзину (кнопка в палитре слоев (внизу слева)).

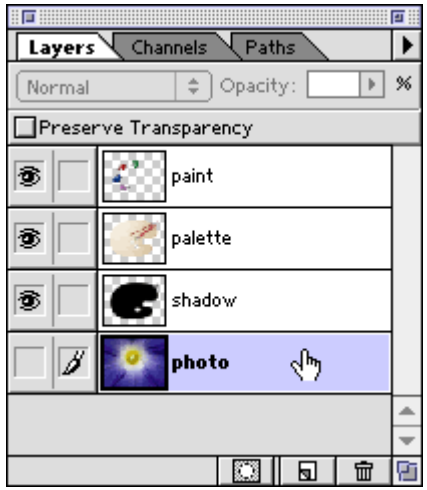


6. Чтобы создать новый слой, нажмите кнопку **New Layer в палитре Layers** . Новый слой появится чуть выше активного слоя.

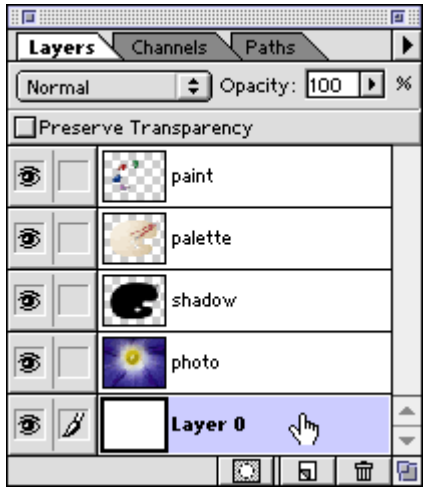




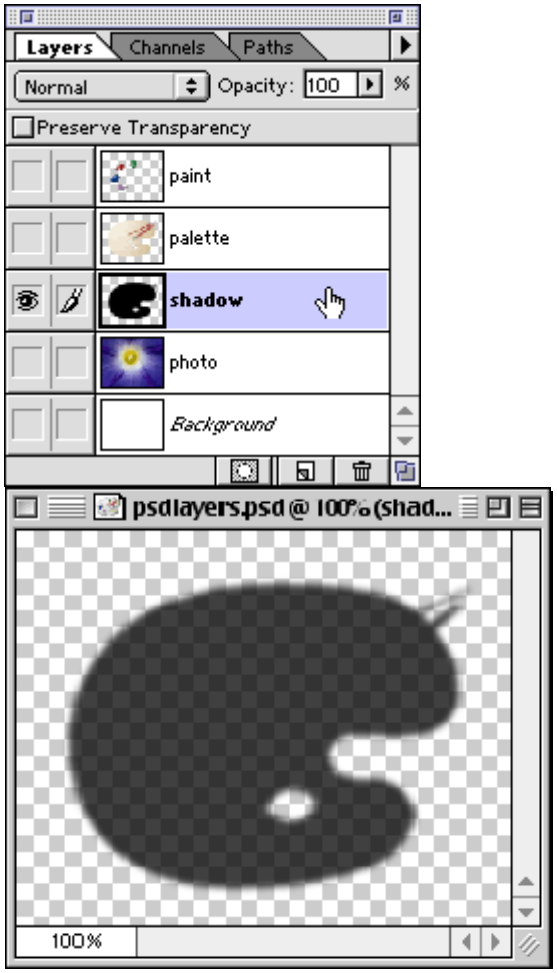
7. Чтобы дублировать слой, перетащите его на кнопку **New Layer** . В приведенном ниже примере, новый слой будет создан с именем *«photo copy»* .



8. Для изменения порядка слоя, нажмите на него и переместите вверх или вниз в **палитре Layers.**



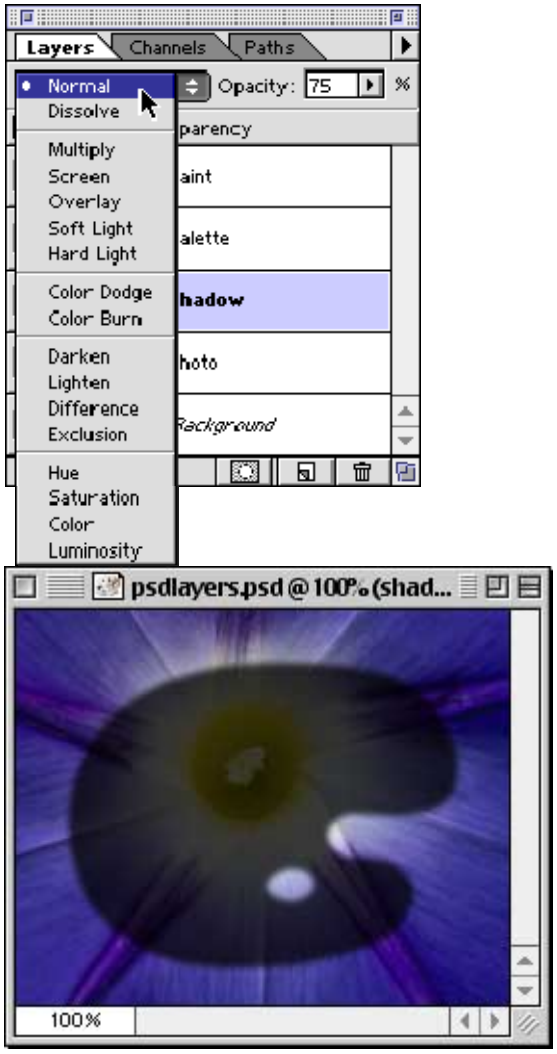
9. Вы можете изменять прозрачность слоя, перетаскивая ползунок **Opacity** . Кроме того, можно просто ввести числовое значение Opacity.



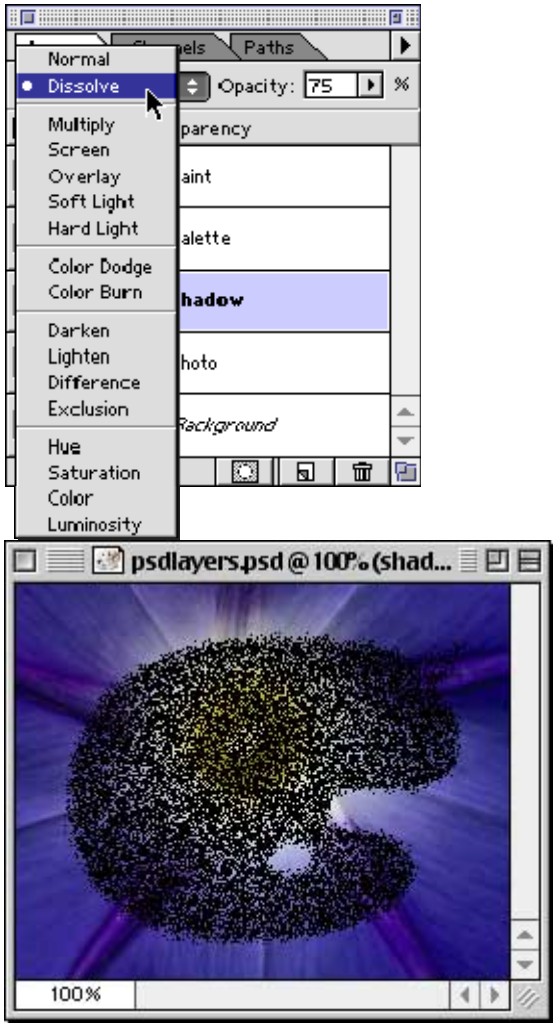
Режимы смешивания слоя

Каждый слой имеет режим смешивания с ниже лежащим.
По умолчанию режим смешивания **«Normal»** . Ниже приведено несколько примеров использования различных видов смешивании:

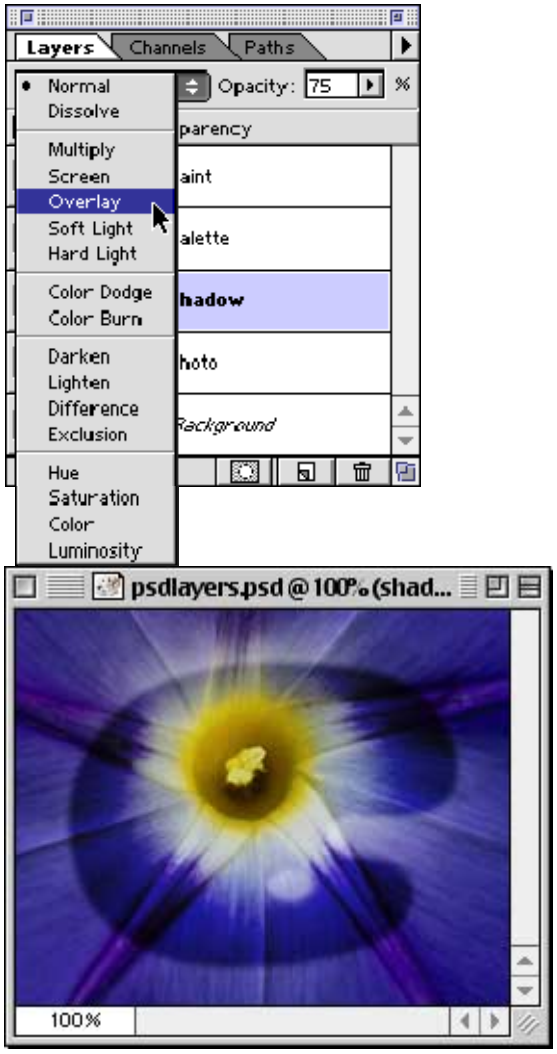
11 a. **Normal**



11 b. **Dissolve**

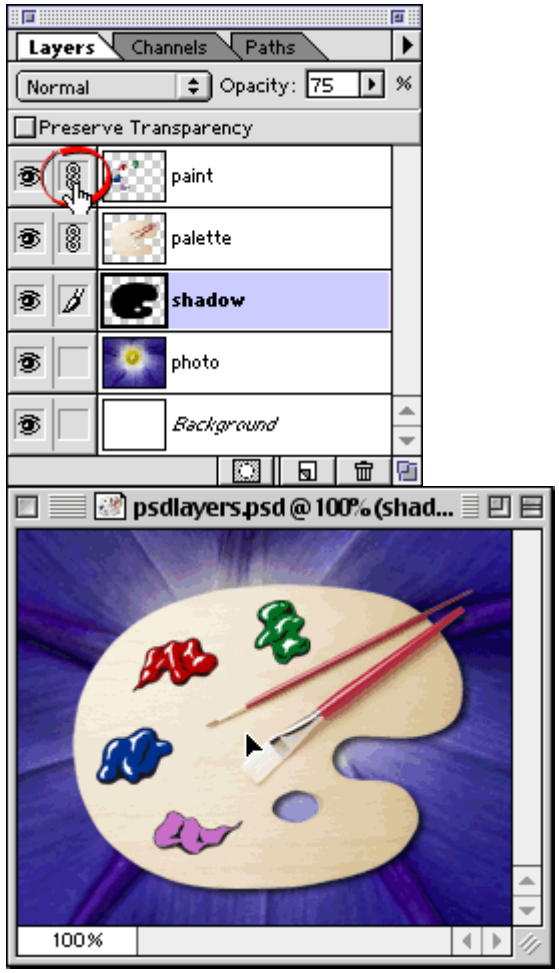


11 c. **Overlay**



Связь слоев

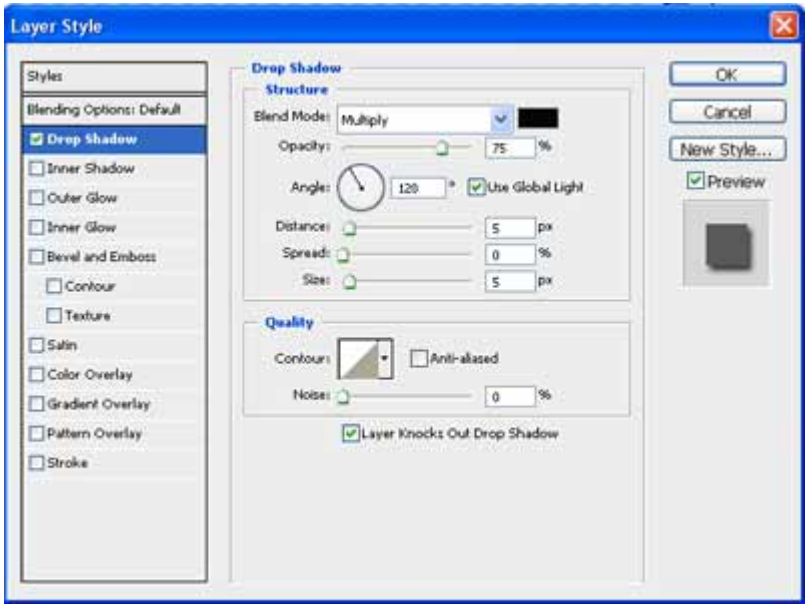
12. Слои могут быть связаны между собой. Нажмите слева от слоя (справа от значка глаза) чтобы связать этот слой с активным (внизу слева). В примере ниже, *«paint», «palette» и «shadow»* , были связаны вместе. При использовании инструмента **Move** , они будут двигаться как единое целое (внизу справа).



Использование стилей слоя в Photoshop

Если вы хотите создать или отредактировать стиль слоя, вам потребуется открыть диалоговое окно Layer Style (Стиль слоя) одним из перечисленных ниже способов. Все выбранные настройки будут применены к текущему слою. Нужно заметить, что невозможно применять стили к фоновому слою.

- Выберите команду Layer>Layer Style (Слой>Стиль слоя) и выберите один из стилей.
- Выберите команду Blending Options (Параметры смешения) в меню палитры Layers.
- Выполните двойной щелчок на эскизе слоя в палитре (если это растровый слой).
- Щелкните на значке Add a layer style (Добавить стиль слоя) в левом нижнем углу палитры и выберите в меню один из стилей или команду Blending Options (Параметры смешения).
- В диалоговом окне Layer Style вы можете выбрать один из эффектов. Для того чтобы в окне появились элементы управления, служащие для настройки определенного эффекта, необходимо первоначально выбрать его в списке слева. Если установлен флажок Preview (Предварительный просмотр), можно свободно экспериментировать, выбирая различные эффекты и меняя их настройки, наблюдая при этом перемены в вашем изображении.



Каждый стиль обладает своим собственным (порою, достаточно сложным) набором настроек. Чтобы выбрать наилучший вариант для

конкретного изображения, нужно просто экспериментировать с различными стилями и настройками, причем последние частично повторяются от стиля к стилю. Поэтому все настройки описаны только для стиля Shadow (Падающая тень), а при описании остальных стилей слоев повторяющиеся настройки опущены.

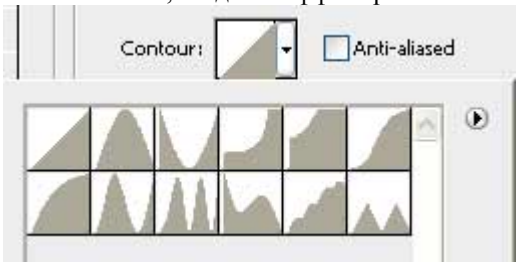
Shadow (Падающая тень)

Каждый материальный объект отбрасывает тень, зависящую от положения источника света. Отбрасывание теней — ключевой элемент в придании изображению реалистичности. Тень повторяет форму изображения, помещенного на текущем слое. Это наиболее простой и быстрый способ придать объемный, реалистичный вид изображению. Для регулировки здесь доступны две группы параметров: Structure (Структура), определяющие непрозрачность, размер, положение и другие параметры, и Quality (Качество), определяющие характер краев тени.

Перечислим эти параметры в порядке их размещения в окне Layer Style.

- Список Blend Mode (Режим смешения) — режимы смешения цвета пикселов тени с нижележащими пикселами изображения.
- Образец цвета, открывающий окно Color Picker (Выбор цвета) — выбор цвета тени.
- Ползунок и поле ввода Opacity (Непрозрачность) — регулировка прозрачности тени.

- Круговой ползунок и поле ввода Angle (Угол) — определяет положение тени относительно объекта.
- Флажок Use Global Light (Использовать глобальный цвет) — включает режим использования текущей установки угла (Angle) для всех эффектов и слоев изображения.
- Ползунок и поле ввода Distance (Расстояние) — определяет смещение тени от объекта.
- Ползунок Spread (Расширение) — определяет степень расширения краевой области тени.
- Ползунок и поле ввода Size (Размер) — определяют ширину области на краю тени, в которой изменяется прозрачность пикселов тени, создавая эффект размытия.



- Раскрывающаяся панель Contour (Контур) — выбор типа контура для создаваемой тени (см. рис. справа).
- Флажок Anti-Alias (Сглаживание) — включает режим сглаживания краевых пикселов тени.
- Ползунок и поле ввода Noise (Шум) — регулировка уровня шума в изображении тени, проявляющаяся в виде появления в ней прозрачных областей.
- Флажок Layer Knocks Out Shadow (Выбивка слоя в области тени) — включение режима выбивки (или вытеснения) для изображения тени.

Inner Shadow (Внутренняя тень)

В отличие от тени, которую слой отбрасывает снаружи себя, внутренняя тень отбрасывается на самом объекте, по направлению от краев к центру. Используйте этот стиль для придания слою эффекта внутренней глубины. Настройки здесь похожи на настройки падающей тени с единственным исключением — поле Choke (Сжать) (степень сжатия краевой области тени) заменяет поле Spread (Расширить), что соответствует внутреннему и внешнему направлениям в распространении тени.

Outer Glow (Внешнее свечение)

Этот стиль идеален для создания «неонового» свечения слоя. Он создает светящийся ореол вокруг границ изображения на слое, при этом размер и характер свечения поддаются широким регулировкам при помощи настройки режима смешения (Blend Mode), непрозрачности (**Opacity**), шума (**Noise**), задания ширины свечения (**Size**), расширения (**Spread**), выбора типа контура свечения (**Contour**).

В области **Structure** (Структура) имеется переключатель, позволяющий выбрать режим раскраски области свечения: градиентом выбранного цвета и прозрачности или градиентом из панели стандартных образцов градиентов.

В списке **Technique** (Техника) области **Elements** (Элементы) можно выбрать вариант формы области свечения. Доступны два варианта: Softer (Мягче) — скругляет форму; Precise(Точный) — повторяет форму объекта.

В области **Quality** (Качество) добавлено еще два параметра.

1. Ползунок и поле ввода **Range** (Диапазон) — определяют область свечения, относящуюся к его контуру (область действия параметра **Contour**).
2. Ползунок и поле ввода **Jitter** (Дрожание) — регулировка уровня случайного смещения (дрожания) пикселов свечения.

Inner Glow (Внутреннее свечение)

Этот эффект аналогичен описанному выше, но с той лишь разницей, что края изображения подсвечиваются изнутри, а не снаружи. При этом ползунок **Choke** (Сжатие) заменяет подобный по смыслу ползунок **Spread** (Расширение) (внутреннее и внешнее направление в распространении свечения).

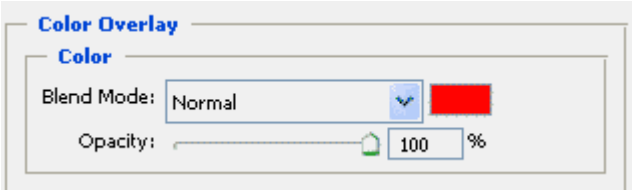
Texture (Текстура)

Для создания эффекта рельефности можете снабдить слой текстурой. Для этого выберите Texture в списке стилей. В результате в окне появятся элементы управления для манипуляций с текстурами. Выберите подходящий узор из раскрывающегося списка (в качестве источника узоров для текстур здесь используются установки **Pattern** (Узор).

В результате к текущему слою будет применена бесцветная текстура, основанная на выбранном узоре. С помощью ползунков Scale (Масштаб) и **Depth** (Глубина) можно настроить масштаб и «глубину» узора. Установка флажка **Invert** (Инвертировать) приведет к «перевороту» рельефа текстуры наоборот.

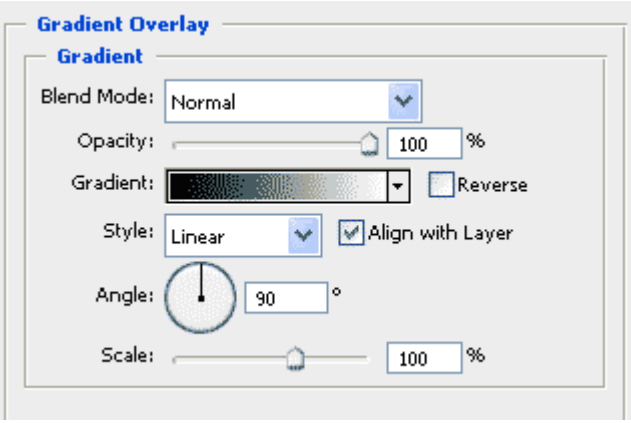
Выбранный и настроенный текстурный стиль можно связать с текущим слоем, установив флажок **Link With Layer** (Связать со слоем). Тогда при перемещении слоя текстура будет также перемещаться.

Color Overlay (Наложение цвета)



Это очень простой стиль — просто все содержимое слоя заполняется выбранным цветом. Остается только выбрать режим смешения и отрегулировать непрозрачность слоя.

Gradient Overlay (Наложение градиента)



Аналогичен наложению цвета, но здесь на пикселы слоя накладывается градиент. Подходящий градиент можно выбрать из списка Gradient (или тут же создать его с помощью редактора градиентов Gradient Editor — открывается двойным щелчком мышью на значке выбранного в списке градиента; стиль градиента — линейный (Linear), радиальный (Radial), угловой (Angle), отраженный (Reflective) или ромбический (Diamond) — из списка Style , масштаб образца градиента — ползунком Scale (Масштаб). Остальные настройки традиционны.

П.2.2. Цветовые модели RGB и CMYK

Основы цветоделения

Человеческий разум на протяжении многих столетий придумывал различные цветовые модели. Уже более трехсот лет назад английский физик Исаак Ньютон открыл, что свет, кажущийся бесцветным, можно с помощью куска стекла (призмы) разложить на множество лучей различного цвета. На этом феномене природы и основываются все цветовые модели, использующие разные составляющие цвета.

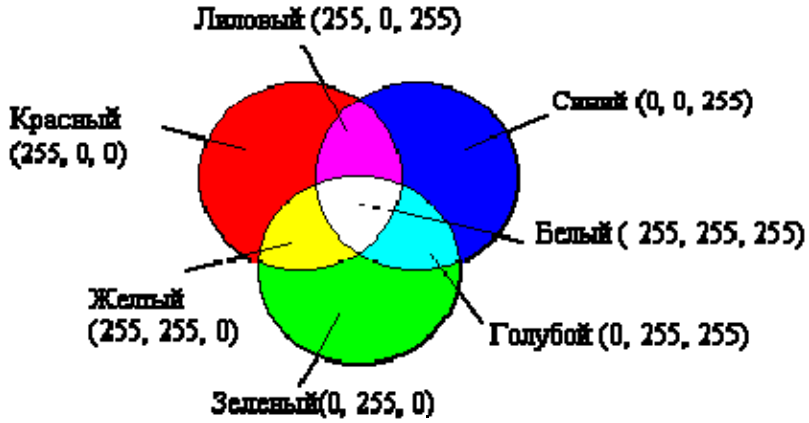
Количество цветов, вообще говоря, бесконечно. Существует, к примеру, стереотип: семь цветов радуги; однако в радуге содержатся ВСЕ цвета: она начинается из инфракрасного диапазона с одного края, и уходит в ультрафиолетовый на другом, содержа все возможные

градации в промежутке (правда, нужно учесть, что каждый оттенок цвета может вдобавок иметь ещё и разную яркость).

Цвета переходят друг в друга, смешиваются и порождают новые. Белый цвет является смешением всех цветов – на них-то белый свет солнца и разлагается в радуге. Существуют так называемые дополнительные цвета – именно их мы видим на цветной негативной плёнке вместо настоящих; если направить на одно и то же место два луча дополнительных цветов, мы получим нейтральный белый – они дополнят друг друга до белого. То же самое произойдёт, если направить в одно место синий, зелёный и красный лучи; собственно, именно это и происходит сейчас: вы читаете это с монитора, на котором цвет каждого пикселя определяется этими тремя составляющими в разных пропорциях.

Способности различных устройств, мониторов ли, принтеров ли, воспроизводить цвет ограничены их техническим устройством. К примеру, принтер печатает на бумаге, и самый яркий цвет, который он может выдать, это сам цвет бумаги, а любая краска, наносимая на неё, будет темнее. У мониторов проблемы состоят в физических свойствах электронно-лучевых трубок либо матриц; у последних, плюс ко всему, до сих пор существует проблема изменения яркости и цвета изображения в зависимости от угла зрения. Есть некоторые цвета, которые можно напечатать, но трудно воспроизвести на мониторе, к примеру напечатанный качественной краской чистый жёлтый, однако чаще всего происходит наоборот – цвет можно показать на мониторе, но нельзя точно напечатать, к примеру очень многие оттенки голубого и синего. Мониторы сами по себе воспроизводят разные диапазоны цветов. Ну и, конечно, сам человеческий глаз некоторые цвета воспринимает лучше других. Чтобы учесть все ньюансы, придумали различные цветовые модели. При отображении изображения на мониторах используется цветовая модель RGB, в полиграфии чаще всего применяют модель CMYK.

Модель RGB



Эта модель описывает излучаемые цвета. Она основана на трех основных (базовых) цветах с длинами волн: 700,0 нм — красный (Red), 546,1 нм -зеленый (Green) и 435,8 нм — синий (Blue). Модель RGB образована от английских и немецких слов: red, rot -красный, green, gran — зеленый, blue, blau — синий, голубой. Остальные цвета получаются сочетанием базовых. Цвета такого смешанного типа называются аддитивными.

Система RGB адекватна цветовому восприятию человеческого глаза, рецепторы которого тоже настроены на красный, зеленый и синий цвета. Остальные цвета воспринимаются как смешение трех основных цветов в различных пропорциях.

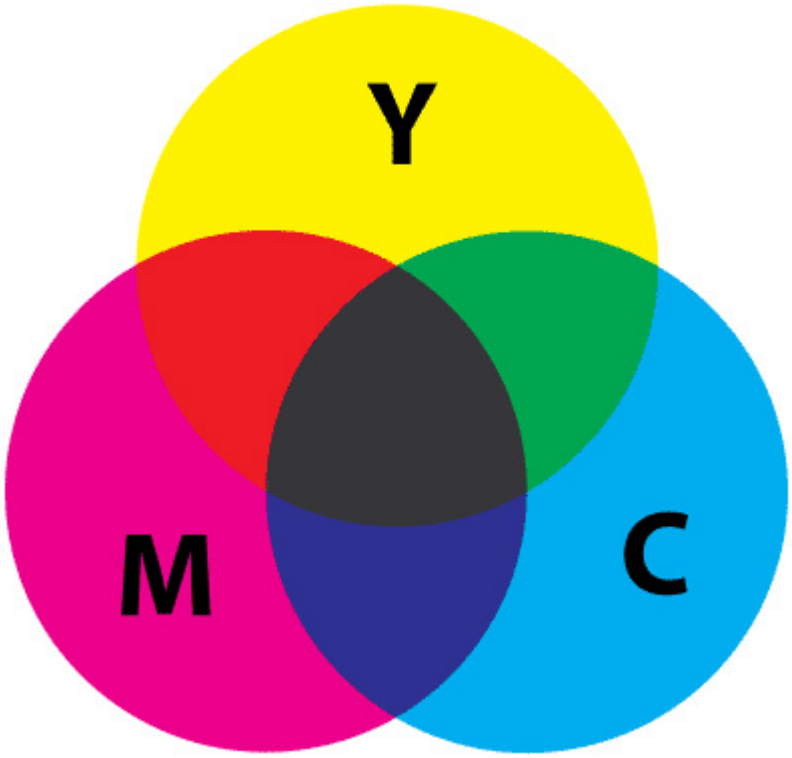
Сочетание зеленого и красного дает желтый цвет, сочетание зеленого и синего — голубой, а сочетание всех трех цветов — белый. В компьютерных технологиях канал изображения кодируется одним байтом. В модели RGB -три канала: красный, синий и зеленый, т. е. RGB — трехканальная цветовая модель. Каждый канал может принимать значения от 0 до 255. Это объясняется тем, что байт состоит из восьми битов, а бит может принимать 2 значения, итого $2^8 = 256$. В RGB, например, у красного цвета может быть 256 градаций: от чисто красного до черного. Таким образом, можно подсчитать, что в модели RGB содержится всего 256^3 или 16 777 216 цветов.

Итак, в соответствии с формулой RGB каждый цвет представлен смешением красного, зелёного и синего с разной яркостью и в разных пропорциях. 256 градаций яркости каждого из них в сумме дают почти 17 миллионов вариантов цветов, покрывая практически весь спектр, который может различить человеческий глаз, и куда более широкий, чем спектр цветов, которые может показать монитор компьютера или напечатать печатающее устройство.

Файл может содержать избыточную информацию, потенциально храня более 65000 градаций каждого из цветов. Больше красок вы от этого не увидите, но расчёты при преобразовании цветов друг в друга в этом случае будут производиться намного точнее, что теоретически благоприятно сказывается на качестве финальной картинки.

Обратите внимание на галерею фотографа начала XX века Прокудина-Горского. Он цветные, хотя цветных фотопластинок тогда не было, не говоря уже о плёнке. Каждый снимок делался на чёрно-белые пластинки три раза, через красный, синий и зелёный фильтры, а потом через эти же фильтры снимки проецировались на экран, создавая полноцветное изображение. Когда мы загружаем RGB изображение в Photoshop, мы можем видеть три цветовых канала – три разных по тональности чёрно-белых изображения, которые создают одно цветное практически так же, как эти чёрно-белые фотопластинки, отснятые через разные фильтры.

Модель CMYK



Рассмотренная в предыдущем разделе модель RGB хорошо описывает цвета, получаемые в результате смешения лучей света различной окраски. Таким образом, она годится для предсказания цветов, видимых на мониторе, а также получающихся при сканировании изображений, но не подходит для печатающих устройств. На бумаге краски смешиваются по-иному: смешение всех цветов даст не белый, а чёрный.

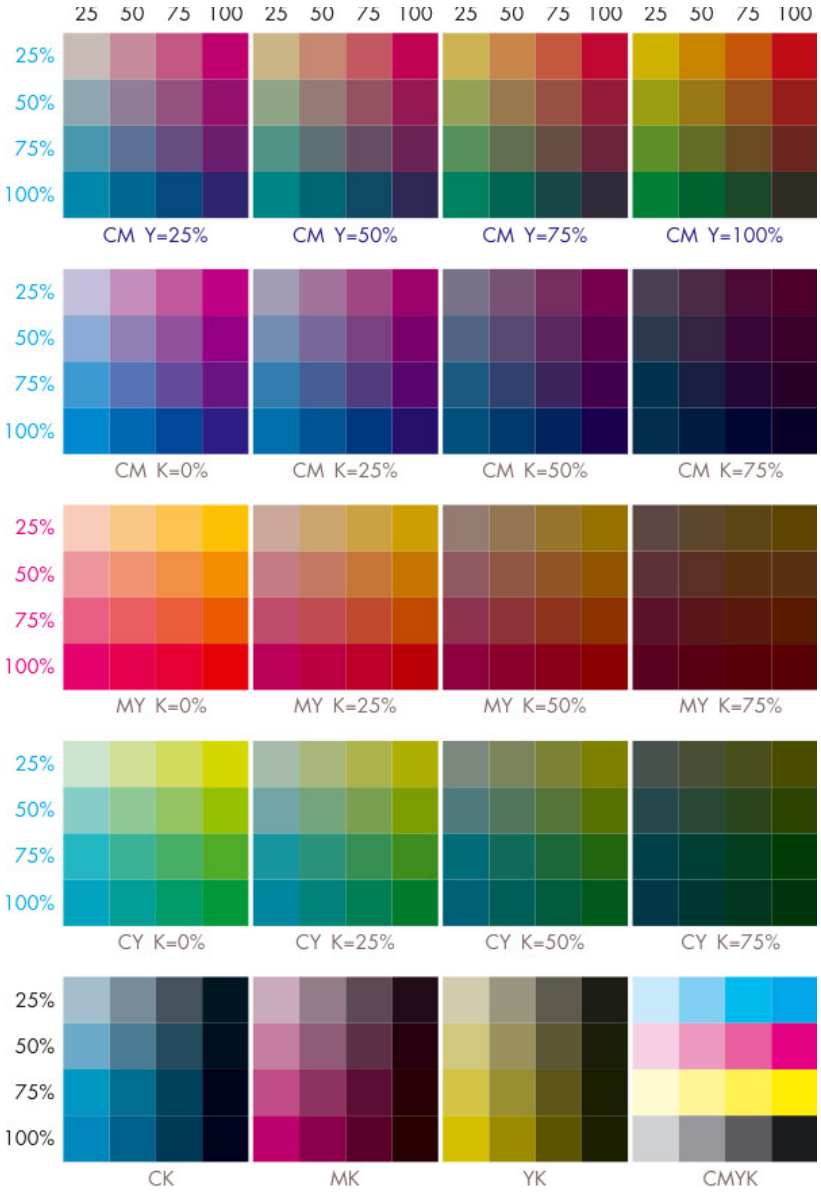
Цветовая модель CMYK является одной из самых популярных моделей, базирующихся на четырех основных цветах принтера. Она является естественным развитием цветовой модели CMY, к которой добавлен черный компонент цвета для получения при печати

действительно черного цвета. В этом случае воспроизведение цветов достигается путем смешивания четырех составляющих: Cyan (Голубая), Magenta (Пурпурная), Yellow (Желтая), blacK (Черная). Интенсивность каждого компонента цвета может изменяться от 0 до 100%. В аббревиатуре этой модели используется буква K для того, чтобы избежать путаницы, поскольку в английском языке с буквы B начинается не только слово Black (Черный), но и слово Blue (Синий).

Есть мнение, что на самом деле CMYK расшифровывается как cyan, magenta, yellow, key color. Причем key color (ключевой цвет) может быть любым: черным, а при желании, к примеру, серебряным.

Таким образом, модель CMYK является четырехканальной. В этом заключается еще одно ее отличие от RGB. Печатающее устройство наносит на бумагу мельчайшие частицы краски, чаще всего одного из четырёх цветов CMYK, и в зависимости от размера и плотности капель каждого цвета мы видим разные цвета и их оттенки.

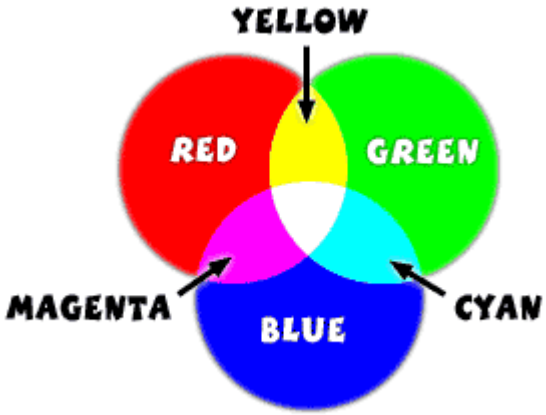
Цветовая модель CMYK описывает поглощаемые цвета. Цвета, которые используют белый свет, вычитая из него определенные участки спектра, называются субтрактивными (вычитаемыми). Именно такие цвета и применяются в модели CMYK. Они получаются путем вычитания из белого аддитивных цветов модели RGB. Голубой цвет получается путем вычитания из белого красного цвета, пурпурный — зеленого, желтый — синего. Если страница, напечатанная в цвете, освещается белым светом, часть света отражается, а часть поглощается красящими пигментами (типографскими красками), при этом глаз воспринимает сочетание основных цветов, которые отражены, а не поглощены. Не весь цвет отражается, часть его все же поглощается пигментом, поэтому отпечатанное на принтере изображение получается менее ярким, чем отображаемое монитором.



На рисунке показана комбинация базовых цветов CMYK

Система CMYK создана и используется для печати. Все файлы, предназначенные для вывода в типографии, должны быть конвертированы в CMYK. Этот процесс называется цветоделением.

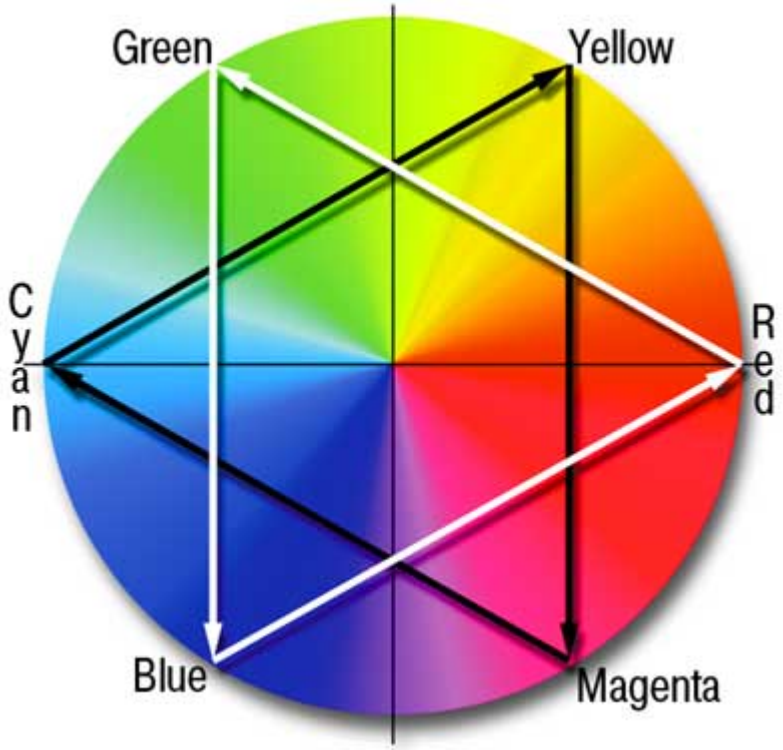
Взаимосвязь основных цветов RGB и CMYK



Каждый из трех базовых цветов модели CMYK получается в результате вычитания из белого цвета одного из базовых цветов модели RGB. Так, например, голубой (cyan) получается вычитанием красного из белого, а желтый (yellow) — вычитанием синего. Напомним, что в модели RGB белый цвет представляется как смесь красного, зеленого и синего максимальной яркости. Тогда базовые цвета модели CMYK можно представить с помощью формул вычитания базовых цветов модели RGB следующим образом:
Cyan = RGB - R = GB = (0,255,255)
Yellow = RGB - B = RG = (255,255,0)
Magenta = RGB - G = RB = (255,0,255)
Заметим, что:
RGB = (0,0,0) - черный цвет
RGB = (255,255,255) - белый цвет
Обратите внимание, что сложение базовых цветов CMYK дает в результате черный: Cyan + Yellow + Magenta = RGB-R-B-G = (0,0,0)
Вычитание из белого всех базовых цветов CMYK дает белый

Вычитание цвета соответствует поглощению его краской. Например, голубая (cyan) краска поглощает из падающего на нее белого света красную составляющую, а все остальное отражает. Этот отраженный

свет наш глаз и воспринимает как голубой. Белый лист бумаги потому кажется нам белым, что он отражает практически весь падающий на него белый свет. С другой стороны, черные предметы почти ничего не отражают, а почти весь свет поглощают. Основные цвета рассмотренных выше моделей RGB и CMYK находятся в зависимости, которую можно представить графически с помощью следующего рисунка:



Каждый цвет расположен напротив дополняющего его и между цветами, с помощью которых он получен. Чтобы усилить какой-либо цвет, необходимо ослабить дополняющий цвет, расположенный на противоположной стороне круга. Например, чтобы усилить желтый (Yellow), надо ослабить синий (Blue). На круге цветов желтый расположен между зеленым (Green) и красным (Red). Сложение этих цветов цветов дает желтый (Yellow).

Следует отметить, что не все цвета модели CMYK могут быть представлены в модели RGB и наоборот. В количественном отношении цветовой диапазон CMYK меньше цветового диапазона RGB. Это обстоятельство имеет принципиальное значение, а не обусловлено только физическими особенностями монитора, печатающего устройства или красок и холста.

Посмотреть и, если надо, поменять цветовые параметры изображения вы всегда сможете, зайдя в подменю Image > Mode (Изображение > Режим). Щелкаете по строке нужной вам цветовой модели, и Photoshop преобразует файл. При переводе многослойного файла Photoshop считает своим долгом спросить вас: а не лучше ли будет сначала слить все слои в один, а потом уже переводить? Варианты ответов: Flatten (соединить слои), Don't Flatten (оставить как есть) и, естественно, Cancel, чтобы отменить перевод. Все операции, какие умеет делать Photoshop, он выполняет для изображений в модели RGB. Файлы, ориентированные непосредственно на полиграфию, нужно перевести в модель CMYK. Естественно, оттенки, выходящие за рамки цветового диапазона CMYK, при таком переводе пропадут, восстановить их уже не удастся. Кроме того, при переводе файла в формат CMYK мы получим файл на треть большего размера, чем он был в RGB: ведь добавляется еще один канал цветовой информации.

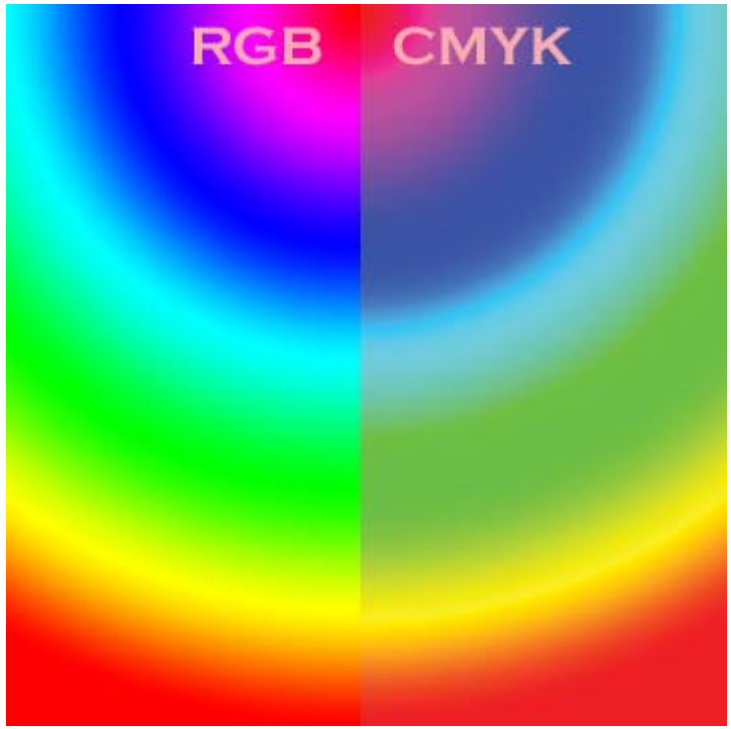
К тому же Photoshop всегда позволит вам разобраться с несоответствием цветовых моделей. Например, работая с рисунком в модели RGB, вы всегда сможете посмотреть, каким он станет в CMYK. Это позволяет сделать команда View > Proof Colors (Просмотр > Цветопроба или Ctrl-Y). Рисунок не переводится в CMYK, а просто показывается так, будто переведен. А в заголовке рисунка появится надпись RGB/CMYK, чтобы мы не забыли, в каком необычном режиме работаем. Есть и другой способ проверки - команда Gamut Warning (Просмотр > Показать цвета вне CMYK, или Ctrl > Shift > Y). По ней Photoshop покажет на рисунке ровным серым фоном области опасных цветов. Сможете оценить, насколько велика «площадь поражения». Так же точно стоит проверять и рисунки, которые вы хотели бы напечатать в типографии. Чтобы потом не удивляться.

Если вы решите вернуться обратно - переделать рисунок из CMYK опять в RGB, утраченные цвета на рисунок, конечно же, не вернутся. Что упало, то пропало. Только отменой команды перевода можно

восстановить исходные цвета. Так что, все эксперименты - только над копиями.

Про формулу CMYK уже говорилось, что она создана под нужды полиграфии. Каждый из четырёх каналов этого пространства определяет, с какой интенсивностью будет наноситься на бумагу одна из четырёх красок.

На рисунке ниже изображено, какизменяются цвета при конверсии из RGB в CMYK:



Как видите, разница налицо, поэтому необходимо внимательно контролировать процесс конверсии, чтобы результат был максимально приближен к оригиналу.

Форматы графических файлов

Итак, на данный момент мы обсудили то, что изображение состоит из пикселей, выяснили откуда они берутся, каким образом могут быть описаны цвет и яркость каждого из них. Осталось обсудить то, как они хранятся, иначе говоря – форматы графических файлов.

Существуют разные форматы графических файлов, все со своими плюсами и минусами. В зависимости от стадии обработки фотографии и конечной цели (различные виды печати, Интернет) предпочтительны разные конечные форматы.

Самым популярным и универсальным форматом является JPEG. Этот формат наиболее часто встречается в Интернете, фото именно в этом формате у вас вероятнее всего примут для печати в фотолаборатории; он разработан специально для фотографий и сходных изображений с большим количеством полутонов. Он позволяет сохранять изображения в пространстве RGB и допускает только 256 градаций яркости каждого цвета – то есть без той дополнительной информации, которая вроде как не видна, но позволяет более качественно преобразовывать изображение; тем не менее, даже имеющейся точности вполне хватает для большинства корректировок фотографий. Кроме того, и это принципиально важно, JPEG это формат, в котором изображение сжимается с потерями. Алгоритм сжатия изображения с помощью сложного математического аппарата записывает данные, из которых потом с некоторой точностью можно восстановить изображение, близкое к исходному. С одной стороны это означает, что объём изображения, т.е. объём файла в килобайтах, можно существенно уменьшить, а с другой то, что чем меньше мы делаем файл, тем сильнее теряем в качестве, и изображение после сохранения не будет полностью соответствовать оригиналу. Звучит страшновато, но на самом деле всё не так плохо.

Во-первых, механизм сжатия JPEG учитывает особенности человеческого зрительного восприятия, поэтому ухудшение качества становится заметным только при весьма существенной степени сжатия. Во-вторых, даже если вы, знакомый с оригиналом, будете видеть какие-то незначительные отличия, другой человек, не видевший оригинала, может даже и не заметить ухудшения качества. В-третьих, при очень сильном сжатии, когда ухудшение качества становится видно и ежу, многие изображения всё равно не теряют своей

наглядности – как большинство снимков экрана в статьях на сайте. Ну и, наконец, в-четвёртых, степень сжатия снимков, которые сохраняются фотокамерой, чаще всего регулируется, и на более высоких установках качества никаких проблем с качеством вы не заметите; этому помогает ещё и крупный размер оригиналов в пикселях: крупные фотографии могут быть сильно уменьшены в объёме без видимых (даже под увеличением более 100%) ухудшений качества.

Простейшим форматом настоящего изображения будет такой, который просто хранит в себе все пиксели, от первого до последнего. Таким форматом является TIFF. Он может хранить пиксели в любом цветовом пространстве, с любой точностью представления. Одно с ним неудобство – файлы получаются очень объёмными; внутреннее архивирование (когда данные внутри файла сжимаются алгоритмом типа zip) практически не уменьшает объём, а только лишь увеличивает требования к вычислительным ресурсам при открытии и сохранении файла.

В итоге: существуют разные форматы но лучше .Tiff для печати пока ничего не придумали, так как это не сжатый формат изображения как JPEG, и для печати где бы то нибыло JPEG нежелателен он "мылит" границы. Из-за того, что JPEG поддерживается повсеместно и позволяет существенно уменьшить объём файлов, это будет лучший формат, для размещения фотографии в Интернете.

Сохранение изображения в формате TIFF

Файл в формате TIFF может быть импортирован многими приложениями, включая QuarkXPress, PageMaker и InDesign (программы верстки). Для этого формата распознаются профили цвета, и возможна установка параметров управления цветом. Программы QuarkXPress, PageMaker и InDesign могут выполнить цветоделение для изображения, имеющего режим CMYK и сохраненного в формате TIFF. Программа Photoshop позволяет сохранять слои в формате TIFF, для этого установите флажок Layers (Слои). Только немногие программы для верстки и графические редакторы могут работать со слоями в формате TIFF, поэтому при импортировании изображения в формате TIFF будет применена операция объединения слоев.

- Выполните команды Image > Mode > CMYK Color (Изображение > Режим > Цвет CMYK), затем в меню File (Файл) воспользуйтесь командой Save As (Сохранить как) или нажмите комбинацию клавиш Ctrl+Shift+S. Введите имя файла и обозначьте папку, в которой он будет сохранен.
- Выберите формат TIFF.
- Щелкните по кнопке Save (Сохранить).
- Если доступны расширенные свойства TIFF-файла, выберите какой-либо метод сжатия в разделе Compression (Сжатие). Все методы сжатия уменьшают размер файла. Однако некоторые программы не могут открыть TIFF-файл, уплотненный с помощью метода JPEG или ZIP. В данном случае рекомендуем использовать метод сжатия LZW; при этом данные изображения не будут утеряны.
- Установите флажок Save Transparency (Сохранить прозрачность), если в изображении есть прозрачные области.
- Щелкните по кнопке OK.
- Программа Photoshop не может сохранить прозрачные области, не сохранив и слои. Поэтому убедитесь, что включены обе опции.

Разрешение изображения

Линейный размер изображения всегда определяется тем, сколько пикселей на единицу измерения способно выдать устройство вывода, будь то монитор или принтер (или сколько вы заставите это устройство выдать). Этот показатель называется разрешающей способностью устройства вывода, и измеряется в пикселях на единицу длины, чаще всего в пикселях на дюйм (dots per inch, dpi). К примеру, у нас есть изображение 900x600 пикселей. При выводе на обычный 17-дюймовый монитор, у которого изображение выставлено на 1024x768 пикселей и который, стало быть, показывает примерно 80 пикселей на дюйм экрана, изображение будет иметь видимые размеры по горизонтали 900/80 = 11.25 дюймов или 28.6 см. Если же печатать это изображение с обычной для качественной печати разрешающей способностью 300 пикселей на дюйм, то изображение будет отпечатано размером 900/300 = 3 дюйма или 7.6 см по горизонтали.

Для печати в типографии снимка в хорошем качестве его разрешение должно быть 300 dpi. Разрешение экрана монитора обычно 72 dpi. Поэтому, фото, добытые из Интернета, обычно, имеют недостаточный размер для печати в хорошем качестве. В этом мы можем убедиться, поменяв его разрешение на 300 dpi и оценив его новый линейный размер.

Для изменения разрешения фотографии, открываем фото и смотрим ее размер Image->Image Size (Изображение->Размер изображения)

Оцениваем длину-ширину в сантиметрах,оцениваем разрешение в dpi. **Чтобы понять реальный размер, снимите галочку с Resample Image и введите значение Resolution - 300 dpi.** Значение ширины-длины изменились. Вот те размеры, которые у вас получились в итоге - это реальные размеры Вашего изображения для печати без потери качества. При этом действии **размеры фотографии в пикселях не изменились**, мы поменяли разрешение фотографии на 300 dpi, подготовив её к печати.

Если увеличивать изображение, изменяя количество пикселей, качество неизбежно ухудшается. Чем конечный размер больше исходного, тем хуже качество. Происходит это потому, что программе приходится "додумывать" недостающие пиксели.

Приложение 3
Библиотека OpenGL

На данный момент в Windows существует два стандарта для работы с трёхмерной графикой: OpenGL, являющийся стандартом де-факто для всех графических рабочих станций, и Direct3D – стандарт, предложенный фирмой Microsoft. Далее будет рассмотрен только стандарт OpenGL

Существенным достоинством OpenGL является его широкая распространенность – он является стандартом в мире графических рабочих станций типа Sun, Silicon Graphics и др. В основу стандарта была положена библиотека IRIS GL, разработанная фирмой Silicon Graphics Inc.

OpenGL представляет собой программный интерфейс к графическому оборудованию (хотя существуют и чисто программные реализации OpenGL). Интерфейс насчитывает около 120 различных команд, которые программист использует для задания объектов и операций над ними (необходимых для написания интерактивных трёхмерны приложений).

OpenGL был разработан как эффективный, аппаратно-независимый интерфейс, пригодный для реализации на различных аппаратных платформах. Поэтому OpenGL не включает в себя никаких специальных команд для работы с окнами или ввода информации от пользователя.

OpenGL позволяет:

1. Создавать объекты из геометрических примитивов (точки, линии, грани и битовые изображения).
2. Располагать объекты в трёхмерном пространстве и выбирать способ и параметры проецирования.
3. Вычислять цвет всех объектов. Цвет может быть как явно задан, так и вычисляться с учётом источников света, параметров освещения, текстур.
4. Переводить математическое описание объектов и связанной с ними информации о цвете в изображение на экране.

При этом OpenGL может осуществлять дополнительные операции, такие, как удаление невидимых фрагментов изображения.

Команды OpenGL реализованы как модель клиент-сервер. Приложение выступает в роли клиента: оно вырабатывает команды, а сервер OpenGL интерпретирует и выполняет их. Сам сервер может находиться как на том же компьютере, на котором находится и клиент, так и на другом.

1.1. Особенности использования OpenGL в Windows

OpenGL представляет собой универсальную графическую библиотеку, которая может быть реализована в любой оконной среде. Поставляется в составе операционной системы Windows, начиная с версии OSR2 в виде двух DLL-файлов – opengl32.dll и glu32.dll. Первая из этих библиотек и есть собственно набор функций OpenGL, вторая содержит дополнительный набор функций, упрощающих кодирование, но построенных и выполняемых с подключением opengl32.dll, и является надстройкой.

То, что эти библиотеки поставляются в составе операционной системы, значительно упрощает распространение разработанных приложений. То, что OpenGL распространяется в виде динамических библиотек, упрощает доступ к его функциям.

Для работы с OpenGL в Windows используется понятие контекста воспроизведения (rendering context), который связывает OpenGL с оконной системой Windows. Если обычный контекст устройства (device context) содержит информацию, относящуюся к графическим компонентам GDI, то контекст воспроизведения содержит информацию, относящуюся к OpenGL.

Таким образом, чтобы начать работать с командами OpenGL, приложение должно создать, как минимум, один контекст воспроизведения и сделать его текущим.

Перед созданием контекста воспроизведения необходимо установить формат пикселей. Для установки формата пикселей используется функция Windows GDI `int ChoosePixelFormat(HDC, const PIXELFORMATDESCRIPTOR)`, выбирающая наиболее подходящий

формат исходя из информации, переданной в полях структуры `PIXELFORMATDESCRIPTOR`.

После того как найден подходящий формат пикселей, следует установить его в контексте устройства при помощи функции `BOOL SetPixelFormat(HDC hDC, inl pixelFormat, const PIXELFORMATDESCRIPTOR)`.

Для работы с контекстом воспроизведения в Windows существуют функции `HGLRC wglCreateContext(HDC hDC)` и `BOOL wglMakeCurrent(HDC hDC, HGLRC hGLRC)`.

Первая из них создаёт новый контекст воспроизведения OpenGL, который подходит для рисования на устройстве, задаваемом контекстом `hDC`. Вторая функция устанавливает текущий контекст воспроизведения.

По окончании работы с OpenGL созданный контекст воспроизведения необходимо удалить. Для этого существует функция `BOOL wglDeleteContext(HGLRC hGLRC)`.

Текущий контекст воспроизведения можно узнать при помощи функции `HGLRC wglGetCurrentContext()`.

При помощи OpenGL можно создавать анимации. При этом для изображения используется режим работы с двумя буферами, когда содержимое одного из них показывается, а в другом осуществляется построение. После окончания построения специальная команда меняет буферы местами (по аналогии с двухстраничным режимом работы). Для использования двойной буферизации необходимо установить флаг `PFD_DOUBLE_BUFFER` при задании формата пикселей и применить команду `SwapBuffers`, меняющую буферы местами (по умолчанию вывод происходит в невидимый буфер).

1.2. Основные типы данных

Все команды (процедуры и функции) OpenGL начинаются с префикса **gl**, а все константы – с префикса **GL_**. Кроме того, в имена функций и процедур OpenGL входят суффиксы, несущие информацию о числе

передаваемых параметров и о их типе. В табл. 1 приводятся вводимые OpenGL типы данных, стандартные типы языка C и суффиксы, которым они соответствуют.

Таблица 1

Типы данных OpenGL

Суффикс	Описание	Тип в C	Тип в OpenGL
b	8-битовое целое	`char`	`GLbyte`
s	16-битовое целое	`short`	`GLshort`
i	32-битовое целое	`long`	`GLint` `GLsizei`
f	32-битовое вещественное число	`float`	`GLfloat`, `GLclampf`
d	64-битовое вещественное число	`double`	`GLdouble`, `GLclampd`
ub	8-битовое беззнаковое целое	`unsigned char`	`GLubyte`, `GLboolean`
us	16-битовое беззнаковое целое	`unsigned short`	`GLushort`
ui	32-битовое беззнаковое целое	`unsigned long`	`GLuint`, `GLenum`, `GLbitfield`
		`void`	`GLvoid`

Некоторые команды OpenGL оканчиваются на букву *v*. Это говорит о том, что команда получает указатель на массив значений, а не сами эти значения в виде отдельных параметров. Многие команды имеют как векторные, так и не векторные версии. Например, конструкции

```
glColor3f(1.0, 1.0, 1.0);
```

и

```
GLfloat color[] = {1.0, 1.0, 1.0};
```



```
glColor3fv(color);
```

эквивалентны.

OpenGL можно рассматривать как автомат, находящийся в одном из нескольких состояний. Внутри OpenGL содержится целый ряд переменных, например, текущий цвет или текущий режим закрашивания. Если установить текущий цвет, то все последующие объекты будут этого цвета до тех пор, пока текущий цвет не будет изменён.

По умолчанию каждая системная переменная имеет своё значение, и в любой момент значение каждой из этих переменных можно узнать. Обычно для этого используется одна из следующих функций: `glGetBooleanv()`, `glGetDoublev()`, `glGetFloatv()` и `glGetIntegerv()`. Для определения значений некоторых переменных служат специальные функции.

OpenGL предоставляет пользователю достаточно мощный, но низкоуровневый набор команд, и все операции высокого уровня должны выполняться в терминах этих команд. Обычно для облегчения работы вместе с OpenGL поставляется библиотека дополнительных команд, каждая из которых начинается с префикса **glu**. В данной лекции будет рассмотрена часть из этих команд.

1.3. Рисование геометрических объектов

1.3.1. Работа с буферами и задание цвета объектов

OpenGL содержит внутри себя несколько различных буферов. Среди них фрейм буфер (где строится изображение), *z*-буфер, служащий для удаления невидимых поверхностей, буфер трафарета и аккумулирующий буфер.

Для очистки внутренних буферов служит процедура `glClear(GLbitfield mask)`, очищающая буферы, заданные переменной `mask`. Параметр mask является комбинацией следующих констант:

GL_COLOR_BUFFER_BIT – очистить буфер изображения (фреймбуфер);

GL_DEPTH_BUFFER_BIT – очистить *z*-буфер;

GL_ACCUM_BUFFER_BIT – очистить аккумулирующий буфер;

GL_STENCIL_BUFFER_BIT – очистить буфер трафарета.

Цвет, которым очищается буфер изображения, задаётся процедурой `glClearColor(GLfloat red, GLfloat green, GLfloat blue, GLfloat alpha)`. Значение, записываемое в *z*-буфер при очистке, задаётся процедурой `glClearDepth(GLfloat depth)`. Значение, записываемое в буфер трафарета при очистке, задаётся процедурой `glClearStencil(GLint s)`. Цвет, записываемый в аккумулирующий буфер при очистке, задаётся процедурой `glClearAccum(GLfloat red, GLfloat green, GLfloat blue, GLfloat alpha)`.

Сама команда `glClear` очищает одновременно все заданные буферы, заполняя их соответствующими значениями.

Для задания цвета объекта служит процедура

```
glColor{3 4}{b s i f d ub us ui}[v](TYPE red, …).
```

Цифра 3 или 4 указывает на количество требуемых аргументов, а буква, следующая за цифрой, показывает тип аргументов. Например, в процедуру `glColor3i` будут переданы три параметра целого типа.

Если значение параметра не задано, то оно автоматически полагается равным единице. Версии процедуры `glColor`, где параметры являются переменными с плавающей точкой, автоматически обрезают переданные значения в отрезок [0, 1].

Процедура `glFlush()` вызывает немедленное рисование ранее переданных команд. При этом ожидания завершения всех ранее

переданных команд не происходит. С другой стороны, команда `glFinish()` ожидает, пока не будут завершены все ранее переданные команды.

Если нужно включить удаление невидимых поверхностей методом *z*-буфера, то *z*-буфер необходимо очистить и передать команду `glEnable(GL_DEPTH_TEST)`. Команду `glEnable()` можно выполнить только один раз при инициализации системных переменных OpenGL. Очистку *z*-буфера необходимо производить перед началом построения очередного кадра изображения.

1.3.2. Задание графических примитивов

Все геометрические примитивы в OpenGL задаются в терминах вершин. Каждая вершина задаётся набором чисел, определяющих её координаты в пространстве.

OpenGL работает с однородными координатами (*x*, *y*, *z*, *w*). Если координата *z* не задана, то она считается равной нулю. Если координата *w* не задана, то она считается равной единице.

Под линией в OpenGL подразумевается отрезок, заданный своими начальной и конечной вершинами.

Под гранью (многоугольником) в OpenGL подразумевается замкнутый выпуклый многоугольник с несамопересекающейся границей.

Все геометрические объекты в OpenGL задаются посредством вершин, а сами вершины задаются процедурой

```
glVertex{2 3 4}{s i f d}[v](TYPE x, …),
```

где реальное количество параметров определяется первым суффиксом (2, 3 или 4), а суффикс *v* означает, что в качестве единственного

аргумента выступает массив, содержащий необходимое количество координат. Например:

```
glVertex2s(1, 2);

glVertex3f(2.3, 1.5, 0.2);

GLdouble vect[] = {1.0, 2.0, 3.0, 4.0};

glVertex4dv(vect);
```

Для задания геометрических примитивов необходимо как-то выделить набор вершин, определяющих этот объект. Для этого служат процедуры `glBegin()` и `glEnd()`. Процедура `glBegin(GLenum mode)` обозначает начало списка вершин, описывающих геометрический примитив. Тип примитива задаётся параметром `mode`, который принимает одно из следующих значений:

`GL_POINTS` – набор отдельных точек;

`GL_LINES` – пары вершин, задающих отдельные точки;

`GL_LINE_STRIP` – незамкнутая ломаная;

`GL_LINE_LOOP` – замкнутая ломаная;

`GL_POLYGON` – простой выпуклый многоугольник;

`GL_TRIANGLES` – тройки вершин, интерпретируемые как вершины отдельных треугольников;

`GL_TRIANGLE_STRIP` – связанная полоса треугольников;

`GL_TRIANGLE_FAN` – веер треугольников;

`GL_QUADS` – четвёрки вершин, задающие выпуклые четырёхугольники;

`GL_QUAD_STRIP` – полоса четырёхугольников.

Процедура `glEnd()` отмечает конец списка вершин.

Между командами `glBegin()` и `glEnd()` могут находится команды задания различных атрибутов вершин: `glVertex()`, `glColor()`, `glNormal()`, `glCallList()`, `glCallLists()`, `glTexCoord()`, `glEdgeFlag()`, `glMaterial()`. Между командами `glBegin()` и `glEnd()` все остальные команды OpenGL недопустимы и приводят к возникновению ошибок.

Рассмотрим в качестве примера задание окружности:

```
glBegin(GL_LINE_LOOP);

for (int i = 0; I < N; i++)

{

float angle = 2 * M_PI * i / N;

glVertex2f(cos(angle), sin(angle));

}

glEnd();
```

Хотя многие команды могут находиться между `glBegin()` и `glEnd()`, вершины генерируются при вызове `glVertex()`. В момент вызова `glVertex()` OpenGL присваивает создаваемой вершине текущий цвет, координаты текстуры, вектор нормали и т. д.

Изначально вектор нормали полагается равным (0, 0, 1), цвет полагается равным (1, 1, 1, 1), координаты текстуры полагаются равными нулю.

1.3.3. Рисование точек, линий и многоугольников

Для задания размеров точки служит процедура `glPointSize(GLfloat size)`, которая устанавливает размер точки в пикселях, по умолчанию он равен единице.

Для задания ширины линии в пикселях служит процедура `gflineWidth(GLfloatwidth)`. Шаблон, которым будет рисоваться линия, можно задать при помощи процедуры `glLineStipple(Glint factor, GLushort pattern)`. Шаблон задается переменной `pattern` и растягивается в `factor` раз. Использование шаблонов линии необходимо разрешить при помощи команды `glEnable(GL_UNE_STIPPLE)`. Запретить использование шаблонов линий можно командой `glDisable(GL_LINE_ST1PPLE)`.

Многоугольники рисуются как заполненные области пикселей внутри границы, хотя их можно рисовать либо только как граничную линию, либо просто как набор граничных вершин.

Многоугольник имеет две стороны (лицевую и нелицевую) и может быть отрисован по-разному, в зависимости от того, какая сторона обращена к наблюдателю. По умолчанию обе стороны рисуются одинаково. Для задания того, как именно следует рисовать переднюю и заднюю стороны многоугольника, служит процедура `glPolygonMode(GLenum face, GLenum mode)`. Параметр `face` может принимать значения `GL_FRONT_AND_BACK` (обе стороны), `GL_FRONT` (лицевая сторона) или `GL_BACK` (нелицевая сторона). Параметр mode может принимать значения `GL_POINT`, `GL_LINE` или `GL_FILL`, обозначая, что многоугольник должен рисоваться как набор граничных точек, граничная ломаная линия или заполненная область, например:

```
glPolygonMode(GL_FRONT, GL_FILL);
```



```
glPolygonMode(GL_BACK, GLJJNE).
```

По умолчанию вершины многоугольника, которые появляются на экране в направлении против часовой стрелки, называются ***лицевыми***. Это можно изменить при помощи процедуры `glFrontFace(GLenum mode)`. По умолчанию параметр mode равняется `GL_CCW`, что соответствует направлению обхода против часовой стрелки. Если задать этот параметр равным `GL_CW`, то лицевыми будут считаться многоугольники с направлением обхода вершин по часовой стрелке.

При помощи процедуры `glCullFace(GLenum mode)` вывод лицевых или нелицевых граней многоугольников можно запретить. Параметр mode принимает одно из значений `GL_FRONT` (оставить только лицевые грани), `GL_BACK` (оставить нелицевые) или `GL_FRONT_AND_BACK` (оставить все грани). Для отсечения граней необходимо разрешить отсечение при помощи команды `glEnable(GL_CULL_FACE)`.

Шаблон для заполнения грани можно задать при помощи процедуры `gIPolygonStipple(const GLubyte * mask)`, где mask задает массив битов размером 32 на 32.

Для разрешения использования шаблонов при выводе многоугольников служит команда `glEnable(GL_POLYGON_STIPPLE)`.

Свой вектор нормали для каждой вершины можно задать при помощи одной из следующих процедур:

```
glNormal3{b s i d f}(TYPE nx, TYPE ny, TYPE nz);

glNormal3{b s I d f}v(const TYPE * v).
```

В версиях с суффиксами *b*, *s* или *i* значения аргументов масштабируются в отрезок [-1, 1].

В качестве примера приведём процедуру, строящую прямоугольный параллелепипед с рёбрами, параллельными координатным осям, по диапазонам изменения *x*, *y* и *z*.

```
#include <windows.h>

#include <gl\gl.h>

drawBox(GLfloat x1, GLfloat x2, GLfloat y1, GLfloat
y2, GLfloat z1, GLfloat

z2)

{

glBegin ( GL_POLYGON );

glNormal3f ( 0.0, 0.0, 1.0 );

glVertex3f ( x1, y1, z2 );

glVertex3f ( x2, y1, z2 );

glVertex3f ( x2, y2, z2 );

glVertex3f ( x1, y2, z2 );

glEnd ();

glBegin ( GL_POLYGON );

glNormal3f ( 0.0, 0.0, -1.0 );

glVertex3f ( x2, y1, z1 );

glVertex3f ( x1, y1, z1 );
```



```
glVertex3f ( x1, y2, z1 );

glVertex3f ( x2, y2, z1 );

glEnd ();

glBegin ( GL_POLYGON );

glNormal3f ( -1.0, 0.0, 0.0 );

glVertex3f ( x1, y1, z1 );

glVertex3f ( x1, y1, z2 );

glVertex3f ( x1, y2, z2 );

glVertex3f ( x1, y2, z1 );

glEnd ();

glBegin ( GL_POLYGON );

glNormal3f ( 1.0, 0.0, 0.0 );

glVertex3f ( x2, y1, z2 );

glVertex3f ( x2, y1, z1 );

glVertex3f ( x2, y2, z1 );

glVertex3f ( x2, y2, z2 );

glEnd ();

glBegin ( GL_POLYGON );

glNormal3f ( 0.0, 1.0, 0.0 );
```

```
glVertex3f ( x1, y2, z2 );

glVertex3f ( x2, y2, z2 );

glVertex3f ( x2, y2, z1 );

glVertex3f ( x1, y2, z1 );

glEnd ();

glBegin ( GL_POLYGON );

glNormal3f ( 0.0, -1.0, 0.0 );

glVertex3f ( x2, y1, z2 );

glVertex3f ( x1, y1, z2 );

glVertex3f ( x1, y1, z1 );

glVertex3f ( x2, y1, z1 );

glEnd ();
```

1.4. Преобразование объектов в пространстве

1.4.1. Преобразования в пространстве

В процессе построения изображения координаты вершин подвергаются определенным преобразованиям. Подобным преобразованиям подвергаются заданные векторы нормали.

Изначально камера находится в начале координат и направлена вдоль отрицательного направления оси *Oz*.

В OpenGL существуют две матрицы, последовательно применяющиеся в преобразовании координат. Одна из них – ***матрица моделирования*** (modelview matrix), а другая – ***матрица проецирования*** (projection matrix). Первая служит для задания положения объекта и его

ориентации, вторая отвечает за выбранный способ проецирования. OpenGL поддерживает два типа проецирования – параллельное и перспективное.

Существует набор различных процедур, умножающих текущую матрицу (моделирования или проецирования) на матрицу выбранного геометрического преобразования.

Текущая матрица задается при помощи процедуры `gIMatrixMode(GLenum mode)`. Параметр mode может принимать значения `GL_MODELVIEW`, `GL_TEXTURE` или `GL_PROJECTION`, позволяя выбирать в качестве текущей матрицы матрицу моделирования (видовую матрицу), матрицу проецирования или матрицу преобразования текстуры.

Процедура `gILoadldenity()` устанавливает единичную текущую матрицу.

Обычно задание соответствующей матрицы начинается с установки единичной и последовательного применения матриц геометрических преобразований.

Преобразование переноса задается процедурой `glTranslate{f d}(TYPE x, TYPE y, TYPE z)`, обеспечивающей перенос объекта на величину (*x*, *y*, *z*).

Преобразование поворота задаётся процедурой `glRotate{f d}(TYPE angle, TYPE x, TYPE y, TYPE z)`, обеспечивающей поворот на угол angle в направлении против часовой стрелки вокруг прямой с направляющим вектором (*x*, *y*, *z*).

Преобразование масштабирования задаётся процедурой `glScale{f d}(TYPE x, TYPE y, TYPE z)`.

Если указано несколько преобразований, то текущая матрица в результате будет последовательно умножена на соответствующие матрицы.

1.4.2. Получение проекций

Видимым объемом при перспективном преобразовании в OpenGL является усеченная пирамида.

Для задания перспективного преобразования в OpenGL служит процедура `gIFrustrum(GLdouble teft, GLdoubte right, GLdouble bottom, GLdouble top, GLdouble near, GLdoubte far).`

Параметры определяют плоскости, по которым проводится отсечение. Величины near и far должны быть неотрицательными.

Иногда для задания перспективного преобразования удобнее воспользоваться следующей процедурой из библиотеки утилит `gluPerspective(GLdouble fovy, GLdouble aspect, GLdouble zNear, GLdouble zFar).`

Эта процедура создает матрицу для задания симметричного поля зрения и умножает текущую матрицу на неё. Здесь `fovy` – угол зрения камеры в плоскости *Oxz*, лежащей в диапазоне [0, 180]. Параметр `aspect` – отношение ширины области к её высоте, `zNear` и `zFar` – расстояния вдоль отрицательного направления оси *Oz*, определяющие ближнюю и дальнюю плоскости отсечения.

Существует ещё одна удобная функция для задания перспективного проецирования `gluLookAt (GLdouble eyeX, GLdouble eyeY, GLdouble eyeZ, GIdouble centerX, GLdouble centerY, GLdouble centerZ, GLdouble upX, GLdouble upY, GLdouble up2).`

Вектор (`eyeX`, `eyeY`, `eyeZ`) задаёт положение наблюдателя, вектор (`cenlerX`, `centerY`, ***`centerZ`***) – направление на центр сцены, а вектор (`upX`, `upY`, `upZ`) – направление вверх.

В случае параллельного проецирования видимым объемом является прямоугольный параллелепипед. Для задания параллельного проецирования служит процедура `glOrtho(GLdouble left,`

`GLdouble right, GLdouble bottom, GLdouble top, GLdouble near, GLdouble far).`

Параметры `left` и `right` определяют координаты левой и правой вертикальных плоскостей отсечения, а `bottom` и `top` – нижней и верхней горизонтальных плоскостей.

Следующим шагом в задании проецирования (после выбора параллельного или перспективного преобразования) является задание области в окне, в которую будет помещено получаемое изображение. Для этого служит процедура `gIViewport(GLint x, GLint y, GLsizei width, GLsizei height).`

Здесь (`x`, `y`) задаёт нижний левый угол прямоугольной области в окне, а `width` и `height` являются её шириной и высотой.

OpenGL содержит стек матриц для каждого из трёх типов преобразований. При этом текущую матрицу можно поместить в стек или извлечь матрицу из стека и сделать её текущей.

Для помещения текущей матрицы в стек служит процедура `gIPushMatrix()`, для извлечения матрицы из стека – процедура `glPopMatrix()`

1.5. Задание моделей закрашивания

Линия или заполненная грань могут быть нарисованы одним цветом (плоское закрашивание, `GL_FLAT`) или путём интерполяции цветов в вершинах (закрашивание Гуро, `GL_SMOOTH`).

Для задания режима закрашивания служит процедура `glShadeModel(GLenum mode)`, где параметр mode принимает значение `GL_SMOOTH` или `GL_FLAT`.

1.6. Освещение

OpenGL использует модель освещённости, в которой свет приходит из нескольких источников, каждый из которых может быть включён или

выключен. Кроме того, существует еще общее фоновое (ambient) освещение.

Для правильного освещения объектов необходимо для каждой грани задать материал, обладающий определенными свойствами. Материал может испускать свой собственный свет, рассеивать падающий свет во всех направлениях (диффузное отражение) или, подобно зеркалу, отражать свет в определенных направлениях.

Пользователь может определить до восьми источников света и их свойства, такие, как цвет, положение и направление. Для задания этих свойств служит процедура `gllight{i f}[v](GLenum light, GLenum pname, TYPE param)`, которая задаёт параметры для источника света `light`, принимающего значения `GL_LIGHT0`, `GL_LIGHT1`, ..., `GL_LIGHT7`. Параметр pname определяет характеристику источника света, которая задается последним параметром.

Для использования источников света расчёт освещенности следует разрешить командой `glEnable(GL_LGHTING)`, а применение соответствующего источника света разрешить (включить) командой `glEnable`, например: `glEnable(GL_LIGHT0)`.

Источник света можно рассматривать как имеющий вполне определенные координаты и светящий во всех направлениях или как направленный источник, находящийся в бесконечно удаленной точке и светящий в заданном направлении (x, y, z).

Если параметр w в команде `GL_POSITION` равен нулю, то соответствующий источник света – направленный и светит в направлении (x, y, z). Если же w отлично от нуля, то это позиционный источник света, находящийся в точке с координатами (x/w, y/w, z/w).

Заданием параметров `GL_SPOT_CUTOFF` и `GL_SPOT_DIRECTION` можно создавать источники света, которые будут иметь коническую направленность. По умолчанию значение параметра `GL_SPOT_CUTOFF` равно 180°, т. е. источник светит во всех направлениях с равной интенсивностью. Параметр `GL_SPOT_CUTOFF` определяет максимальный угол от направления источника, в котором

распространяется свет от него. Он может принимать значение 180° (не конический источник) или от 0 до 90°.

Интенсивность источника с расстоянием, вообще говоря, убывает (параметры этого убывания задаются при помощи параметров `GL_CONSTANT_ATTENUATION`, `GL_LINEAR_ATTENUATION` и `GL_QUADRATIC_ATTENUATION`). Только собственное свечение материала и глобальная фоновая освещенность с расстоянием не ослабевают.

Глобальное фоновое освещение можно задать при помощи команды `gtLightModel{i f} [v] (GL_LIGHT_MODEL_AMBIENT ambientColor)`.

Местонахождение наблюдателя оказывает влияние на блики на объектах. По умолчанию при расчётах освещённости считается, что наблюдатель находится в бесконечно удалённой точке, т. е. направление на наблюдателя постоянно для любой вершины. Можно включить более реалистическое освещение, когда направление на наблюдателя будет вычисляться для каждой вершины отдельно. Для этого служит команда `glLightModeli(GL_LIGHT_MODEL_LOCAL_VIEWER, GL_TRUE)`.

Для задания освещения как лицевых, так и нелицевых граней (для нелицевых граней вектор нормали переворачивается) служит следующая команда `glLightModeli(GL_LIGHT_MODEL_TWO_SIDE, GL_TRUE)`, причём существует возможность отдельного задания свойств материала для каждой из сторон.

Свойства материала, из которого сделан объект, задаются при помощи процедуры `glMaterial{i f}[v](GLenum face, GLenum pname, TYPE param)`.

Параметр `face` указывает, для какой из сторон грани задается свойство, и принимает одно из следующих значений: `GL_BACK`, `GL_FRONT_AND_BACK`, `GL_FRONT`.

Параметр `pname` указывает, какое именно свойство материала задается.

Расчёт освещённости в OpenGL не учитывает затенения одних объектов другими.

1.7. Полупрозрачность. Использование α-канала

До сих пор не рассматривался α-канал (в RGBA-представлении цвета) и значение соответствующей компоненты во всех примерах всегда равнялось единице. Задавая значения, отличные от единицы, можно смешивать цвет выводимого пикселя с цветом пикселя, уже находящегося в соответствующем месте на экране, создавая тем самым эффект прозрачности.

При этом наиболее естественно думать об этом, считая что RGB-компоненты задают цвет фрагмента, α-значение – его непрозрачность (степень поглощения фрагментом проходящего через него света). Так, если у стекла установить значение, равное 0.2, то в результате вывода цвет получившегося фрагмента будет на 20 % состоять из собственного цвета стекла и на 80 % – из цвета фрагмента под ним.

Для использования α-канала необходимо сначала разрешить режим прозрачности и смешения цветов командой `gEnable(GL_BLEND)`.

В процессе смешения цветов цветовые компоненты выводимого фрагмента $R_sG_sB_sA_s$ смешиваются с цветовыми компонентами уже выведенного фрагмента $R_dG_dB_dA_d$ по формуле

$(R_sS_r+R_dD_r,\ G_sS_g+G_dD_g,\ B_sS_b+B_dD_b,\ A_sS_a+A_dD_a)$,

где (S_r, S_g, S_b, S_a) и (D_r, D_g, D_b, D_a) – коэффициенты смешения.

Для задания связи этих коэффициентов с α-значениями используется функция `glBlendFunc(GLenum sfactor, GLenum dfactor)`.

Здесь параметр `sfactor` задаёт то, как нужно вычислять коэффииенты (S_r, S_g, S_b, S_a), а параметр `dfactor` – коэффициенты (D_r, D_g, D_b, D_a).

1.8. Наложение текстуры

Текстурирование позволяет наложить изображение на многоугольник и вывести этот многоугольник с наложенной на него текстурой, соответствующим образом преобразованной. OpenGL поддерживает одно- и двумерные текстуры и различные способы наложения (применения) текстуры.

Для использования текстуры надо сначала разрешить одно- или двумерное текстурирование при помощи команд `glEnable(GL_TEXTURE1D)` или `glEnable(GL_TEXTURE_2D)`.

Для задания двумерной текстуры служит процедура
```
glTexImage2D(GLenum target, GLint level, GLint
   component, GLsizei width, GLsizei height, GLint
  border, GLenum format, GLenum type, const GLvoid
*pixels).
```

Параметр `target` зарезервирован для будущего использования и в текущей версии OpenGL должен быть равен `GL_TEXTURE_2D`. Параметр `level` используется в том случае, если задается несколько разрешений данной текстуры. При ровно одном разрешении он должен быть равным нулю.

Следующий параметр – `component` – целое число от одного до четырех, показывающее, какие из RGBA-компонент выбраны для использования. Значение 1 выбирает компоненту *R*, значение 2 выбирает *R* и *A* компоненты, 3 соответствует *R*, *G* и *B*, а 4 соответствует компонентам RGBA.

Параметры `width` и `height` задают размеры текстуры, `border` задает размер границы (бортика), обычно равный нулю. Как параметр `width`, так и параметр `height`, должны иметь вид $2^n + 2b$, где n – целое число, а b – значение параметра `border`. Максимальный размер текстуры зависит от реализации OpenGL, но он не менее 64 на 64.

При текстурировании OpenGL поддерживает использование пирамидального фильтрования (mip-mappping). Для этого необходимо иметь текстуры всех промежуточных размеров, являющихся степенями двух, вплоть до 1×1, и для каждого такого разрешения вызвать

glTexImage2D с соответствующими параметрами level, width, height и image. Кроме того, необходимо задать способ фильтрования, который будет применяться при выводе текстуры.

Под фильтрованием здесь подразумевается способ, которым для каждого пикселя будет выбираться подходящий элемент текстуры (тексель). При текстурировании возможна ситуация, когда одному пикселю соответствует небольшой фрагмент текселя (увеличение) или же, наоборот, когда одному пикселю соответствует целая группа текселей (уменьшение).

Способ выбора соответствующего текселя, как для увеличения, так и для уменьшения (сжатия) текстуры необходимо задать отдельно. Для этого используется процедура glTexParameteri(GL_TEXTURE_2D, GLenum p1, GLenum p2), где параметр p1 показывает, задается ли фильтр для сжатия или для растяжения текстуры, принимая значение GL_TEXTURE_MIN_FLITER или GL_TEXTURE_MAG_FILTER. Параметр p2 задает способ фильтрования.

При использовании пирамидального фильтрования помимо выбора текселя на одном слое текстуры появляется возможность либо выбрать один соответствующий слой, либо проинтерполировать результаты выбора между двумя соседними слоями. Для правильного применения текстуры каждой вершине следует задать соответствующие ей координаты текстуры при помощи процедуры glTexCoord{1 2 3 4}{s i f d}[v](TYPE coord, ...).

Этот вызов задаёт значения индексов текстуры для последующей команды glVertex.

Если размер грани больше, чем размер текстуры, то для циклического повторения текстуры служат команды glTexParameteri (GL_TEXTURE_2D, GL_TEXTURE_S_WRAP, GL_REPEAT), glTexParameteri(GL_TEXTURE_2D, GL_TEXTURE_T_WRAP, GL_REPEAT).

Координаты текстуры обычно подвергаются преобразованию при помощи матрицы текстурирования. По умолчанию она совпадает с

единичной матрицей, но пользователь сам имеет возможность задать преобразования текстуры, например следующим образом:

```
glMatrixMode(GL_TEXTURE);

glRotatef(...};

glMatrixMode(GL_MODELV1EW);
```

При выводе текстуры OpenGL может использовать линейную интерполяцию (аффинное текстурирование) или же точно учитывать перспективное искажение. Для задания точного текстурирования служит команда glHint(GL_PERSPECTIVE_CORRECTION_HINT, GL_NICEST).

Если качество не играет большой роли, а нужна высокая скорость рендеринга, то в качестве последнего аргумента следует использовать константу GL_FASTEST.

Описанный выше способ работ с текстурами используется в OpenGL версии 1.0. В более новых версиях OpenGL, начиная с версии 1.1, введены дополнительные функции, повышающие удобство работы с текстурами. В OpenGL 1.0 процедуру glTexImage2D необходимо вызывать всякий раз, когда нужно сменить текущую текстуру. Это достаточно медленный способ. В OpenGL 1.1 имеется возможность присваивать имена текстурам и затем сменять текущую текстуру только указанием имени новой текстуры, без повторной её загрузки в память процедурой glTexImage2D.

Имя текстуры представляет собой уникальное значение типа GLuint. Перед использованием текстуры необходимо присвоить ей имя. Имена текстур можно сгенерировать при помощи процедуры glGenTextures(GLsizei n, GLuint *textures).

Параметр n определяет количество текстур, для которых необходимо сгенерировать имена. Параметр textures является указателем на массив переменных типа GLuint, состоящим из *n* элементов. После вызова процедуры каждый элемент массива будет содержать уникальное имя текстуры, которое затем может быть использовано при работе с текстурами.

Для выбора текущей (активной) текстуры используется функция `glBindTexture(GLenum target, GLuint texture)`.

Параметр `target` определяет тип текстуры (одномерная – `GL_TEXTURE_1D` или двумерная – `GL_TEXTURE_2D`). На практике более часто используются двумерные текстуры, которые представляют собой обычные двумерные изображения. Параметр `texture` определяет имя текстуры, которую необходимо сделать активной.

После того, как установлена активная текстура, можно вызвать процедуру `glTexImage2D` и задать параметры текстуры, а также сами её тексели. После вызова процедуры `glTexImage2D` текстура готова к применению.

Для того чтобы наложить текстуру на объект или многоугольник достаточно установить активную текстуру (процедура `glBindTexture`) и определить текстурные координаты при помощи процедуры `glTexCoord`.

Достоинство использования функций OpenGL 1.1 для работы с текстурами заключается не только в более высоком быстродействии по сравнению с использованием процедуры `glTexImage2D`, но и в повышенном удобстве работы с текстурами. Создав массив текстурных имён можно работать одновременно с несколькими текстурами, вызывая лишь функцию `glBindTexture` по мере необходимости.

2. Аппаратные средства машинной графики

Математическое и программное обеспечение компьютерной графики нельзя рассматривать в отрыве от аппаратных средств, применяемых на различных этапах работы с изображениями. Все эти средства принято делить на три большие группы:

- устройства ввода (сканеры, дигитайзеры/графические планшеты, цифровые фото- и видеокамеры);
- устройства вывода (мониторы, принтеры, плоттеры, цифровые проекторы);
- устройства обработки (графические ускорители, кодеры MPEG и др.).

Поскольку детальная информация о принципах действия, параметрах и применении вышеперечисленных устройств дается в соответствующих разделах других курсов, например «Периферийные устройства ЭВМ», подробнее остановимся только на аппаратных средствах первой группы, на рынке которых наблюдается в настоящий момент бум новых средств и технологий.

2.1. Устройства ввода

Существуют различные технические средства, осуществляющие процесс преобразования изображений в цифровую форму, например: сканеры, дигитайзеры (графические планшеты), цифровые фото- и видеокамеры. В каждом конкретном случае важно правильно выбрать нужное устройство, руководствуясь его техническими характеристиками, для получения оцифрованного изображения с требуемой детальностью и цветовой гаммой.

2.1.1. СКАНЕРЫ

Сканером называется устройство, позволяющее вводить в компьютер образы изображений, представленных в виде текста, рисунков, слайдов, фотографий или другой графической информации.

Традиционно сканеры служили для решения специализированных задач: ввода и запоминания изображений в настольных издательских системах, организации хранения текстовых документов в юридических фирмах и т. п. С появлением почти у каждого собственных страниц Web и повсеместным распространением цветных струйных принтеров сканеры быстро превращаются в универсальные настольные средства подобно принтерам и модемам.

Принцип действия и виды сканеров

Принцип действия практически всех типов сканеров един. Он основан на том, что направленным лучом освещаются отдельные точки исходного изображения (оригинала) и отраженный в результате луч воспринимается фоточувствительным приемником, где информация о «цвете» точки интерпретируется как конкретное численное значение, которое через определенный интерфейс передается в компьютер.

Как правило, светочувствительные элементы объединяют в матрицу, для того, чтобы сканировать одновременно целый участок оригинала.

По механизму перемещения матрицы светочувствительных элементов относительно оригинала выделяют следующие типы сканеров:

1) ***Планшетный сканер*** (Flatbed Scanner) – сканер, в котором оригинал кладется на стекло и сканируется при помощи подвижной линейной матрицы (рис 2.1). Размеры матрицы и системы фокусировки подобраны так, чтобы вести сканирование листа по всей ширине.

Рис. 2.1. Планшетный сканер

2) ***Ручной сканер*** (Handheld Scanner) – портативный сканер, в котором сканирование осуществляется путем ручного перемещения сканера по оригиналу. По принципу действия такой сканер аналогичен планшетному. Ширина области сканирования не более 15 см.

3) ***Барабанный сканер*** (Drum Scanner) – сканер, в котором оригинал закрепляется на вращающемся барабане. При этом сканируется точечная область изображения, а сканирующая головка движется вдоль барабана на очень маленьком расстоянии от оригинала.

Основные характеристики

При выборе конкретной модели сканера необходимо учитывать ряд характеристик, связанных с техническими возможностями модели.

Разрешение (Resolution) – число точек или растровых ячеек, из которых формируется изображение, на единицу длины или площади. Чем больше разрешение устройства, тем более мелкие детали могут быть воспроизведены.

Аппаратное/оптическое разрешение сканера (Hardware/optical Resolution) – это одна из основных характеристик сканера, напрямую связанная с плотностью размещения чувствительных элементов на матрице сканера. Измеряется в количестве пикселов на квадратный дюйм изображения – PPI (Pixel Per Inch). Пример: 300×300 ppi.

Интерполированное разрешение (Interpolated Resolution) – разрешение изображения, полученного при помощи математической обработки исходного изображения. С улучшением качества имеет мало общего. Часто служит рекламной уловкой для неподготовленных пользователей. Пример: 600×1200 (9600) ppi (цифра 600 – максимальное оптическое разрешение, 1200 – разрешение "двойного шага", 9600 – максимальное интерполированное разрешение).

Глубина цвета (color depth) – количество разрядов каждого пиксела в цифровом изображении, в том числе выдаваемом сканером. Описывает максимальное количество цветов, воспроизводимое сканером в виде степени числа 2. Одному разряду соответствует черно-белое изображение, 8-ми – серое полутоновое, 16-ти – цветное, 24-цветное изображение – наиболее близкое к человеческому восприятию (модель

RGB), 36bit и больше – полноцветное изображение с высокой достоверностью цветопередачи, предназначенное для профессиональной работы, чаще всего в издательском деле.

Фирмы-производители

На мировом рынке представлено достаточно большое число фирм-производителей сканеров. Наиболее популярные модели производят Hewlett-Packard, Agfa, Canon, Mustek.

2.1.2. ДИГИТАЙЗЕРЫ

Дигитайзер (графический планшет) — это устройство, предназначенное для оцифровки изображений, применяемое для создания на компьютере рисунков и набросков (рис 8.2). Художник создает изображение на экране, но его рука водит пером по планшету. Как правило, планшет используют профессиональные художники для более точной обработки (создания) изображений.

Рис. 2.2. Графический планшет

Кроме того, дигитайзер можно использовать просто как аналог манипулятора «мышь».

Принцип действия

Дигитайзер, или планшет, как его часто называют, состоит из двух основных элементов: основания и курсора, двигающегося по его поверхности. Принцип действия дигитайзера основан на фиксации местоположения курсора с помощью встроенной в планшет сетки. При нажатии на кнопку курсора его местоположение на поверхности планшета фиксируется, а его координаты передаются в компьютер. Сетка состоит из проволочных или печатных проводников с довольно большим расстоянием между соседними проводниками (от трех до шести мм).

Основные характеристики

Механизм регистрации позволяет получить шаг считывания информации намного меньше шага сетки (до 100 линий на мм). Шаг считывания информации называется ***разрешением дигитайзера***.

Шаг считывания регистрирующей сетки является физическим пределом разрешения дигитайзера. Мы говорим о пределе разрешения, потому что следует различать разрешение как характеристику прибора и ***программно-задаваемое разрешение***, что есть переменная величина в настройке дигитайзера.

Следует отметить, что в работе планшетов возможны помехи со стороны излучающих устройств, в частности мониторов. Независимо от принципа регистрации существует погрешность в определении координат курсора, именуемая ***точностью дигитайзера***. Эта величина зависит от типа дигитайзера и от конструкции его составляющих. Точность существующих планшетов колеблется в пределах от 0.005 дюйма до 0.03 дюйма.

Важной характеристикой дигитайзера является регистрируемое ***число степеней нажатия*** электронного пера. В существующих моделях эта величина может изменяться в пределах от одного до 256-ти. Программа-обработчик использует эту величину, устанавливая в зависимости от нее, например, толщину проводимой линии (чем сильнее нажим, тем толще линия).

Фирмы-производители

Наиболее популярны модели следующих фирм: CalComp, NUMONICS, WACOM.

2.1.3. ЦИФРОВЫЕ ФОТОКАМЕРЫ

Цифровая камера получает изображения, обрабатывает их и хранит в цифровом формате. Вместо пленки она использует встроенную или сменную полупроводниковую память, чтобы хранить снимки. Она обладает теми же основными свойствами, что и нормальная фотокамера, и, помимо этого, может соединяться с компьютером, телевизором или принтером. Поскольку обработка кадра происходит непосредственно в камере, пользователь может сразу проверить правильность полученного изображения, напечатать его или послать по электронной почте.

Рис. 2.3. Цифровая фотокамера

Достоинства цифровых фотокамер:

- Изображение обрабатывается сразу же. Большинство цифровых камер имеют маленький цветной экран, на котором можно немедленно увидеть снимок, который был сделан. Это позволяет отказаться от неудачного кадра и записать на его место другой.

- Изображения хранятся в электронной памяти, циклы записи-стирания информации в которую могут повторяться

практически бесконечно. Пропадает необходимость каждый раз покупать пленку, реактивы для ее проявки.

- Упростился процесс ввода фотографий в компьютер. Теперь не нужно сканировать изготовленные обычным образом фотографии. Вы просто подключаете цифровую камеру с помощью кабеля или PC-карты к ПК и переписываете нужные снимки на жесткий диск.

- Цифровая камера позволяет проводить множество манипуляций с фотографиями.

Недостатки цифровых фотокамер:

- Низкое разрешение. Приемлемым для качественной печати разрешением (свыше 300 dpi) обладают на сегодняшний день только профессиональные цифровые камеры со стоимостью, делающей их недоступными для массового пользователя.

- Высокая, по сравнению с обычными фотокамерами такого же класса, цена.

- Действительно, качественная печать цифровых фотографий требует чаще всего специального оборудования и имеет высокую себестоимость за счет дорогих расходных материалов.

Принцип действия

Принцип действия цифровой фотокамеры аналогичен принципу действия видеокамеры и состоит в следующем. Пучок лучей света от объекта съемки, проходя через линзу (или систему линз) объектива и диафрагму, попадает на матрицу CCD (Charged Coupled Device). Матрица CCD или, как ее еще называют, ПЗС (преобразователь свет-сигнал) представляет собой прямоугольную матрицу из светочувствительных элементов. Луч света, попадая на чувствительный элемент, преобразуется в аналоговый электрический сигнал. Аналоговые сигналы от CCD преобразуются в цифровые, обрабатываются и записываются в память. Преобразование сигналов в цифровую форму производится с помощью аналого-цифрового преобразователя ADC.

Кроме CCD, ADC и памяти в электрическую схему цифровой фотокамеры входят процессор DSP, который формирует изображение из цифровых потоков, и конвертор JPEG, сжимающий изображения для увеличения количества хранимых кадров.

Сменная память используется в цифровых камерах для увеличения количества сохраняемых кадров и, чаще всего, представляет собой Flash-карту памяти.

Фирмы-производители

В настоящее время на рынке работают десятки известнейших фирм-производителей как традиционного фотооборудования и материалов (Kodak, Konica, Nikon, Fuji, Agfa, Olympus и др.), так и компьютерной периферии и прочего электронного оборудования (Hewlett-Packard, Seiko Epson, Sony, Ricoh, Mustek, UMAX, LG Electronics, Minolta и др.).

ЛИТЕРАТУРА

1. В.Ф.Асмус. Проблема интуиции в философии и математике. «Мысль», М.1965.

2. Дж. Барвайс. Введение в логику первого порядка. Справочная книга по математической логике. Ч.1. «Наука», М.1982. Пер. с англ..: Handbook of mathematical logic. J. Barwise (Ed). North-Holland P.C. 1977.

3. Дж.Булос, Р.Джеффри. Вычислимость и логика. М. «Мир» 1994. Пер. с английского: George S. Boolos, Richard C. Jeffrey. Computability and logic. Cambridge University press, 1989.

4. Е.А.Беляев, В.Я.Перминов. Философские и методологические проблемы математики. Изд-во Московского университета, 1981.

5. М.Бунге. Интуиция и наука. «Прогресс», М. 1967. Пер. с английского: M.Bunge. Intuition and Science. New York, 1962.

6. Н.Бурбаки. Начала математики. Ч.1, кн.1. Теория множеств. Мир. М. 1965. Пер. с французского.: Elements de Mathematique par N.Bourbaki. Livre 1. Theorie des ensembles. Troisieme edition, 1958.

7. Е.Вигнер. Непостижимая эффективность математики в естественных науках. УФН, т.94, вып.3, 1968, 535 – 546. Пер. с англ.: E.Wigner. The Unreasonable Effectiveness of Mathematics in the Natural Sciences, Comm. Pure and Appl. Math. 131, 1 (1960).

8. Р.Декарт. Правила для руководства ума. Избранные произведения. М.1950. Пер. с французского: Descartes R. Oeuvres, t. X. Paris, 1908.

9. М.Клайн. Математика. Утрата определённости. М.”Мир”, 1984. Пер. с англ.: Morris Kline. MATHEMATICS. The Loss of Certainty. N-Y, Oxford University Press, 1980.

10. С.К.Клини. Введение в метаматематику. ИЛ М. 1957. Пер.с англ. : Introduction tu metamathematics by Stephen Cole Kleene. 1952. D.van

Nostrand Company, inc. New York , Toronto.

11. М.Кац, С.Улам. Математика и логика. Ретроспектива и перспективы. «Мир», М. 1971. Пер. с англ.: Mathematics and Logic. Retrospect and Prospects. Mark Kac and Stanislaw M. Ulam. N.-Y. Washington. London. 1968.

12. А.Е. Кононюк. Общая теория познания и созидания. Кн.1. Киев: «Освіта України», 2013. 648 с. *ecat.diit.edu.ua:81/ft/index_ru.html*
13. А.Е. Кононюк. Общая теория познания и созидания. Кн.2, ч.1. Киев: «Освіта України», 2013. 544 с.
ecat.diit.edu.ua:81/ft/index_ru.html
14. А.Е. Кононюк. Общая теория познания и созидания. Кн.2, ч.2. Киев: «Освіта України», 2013. 644 с.
ecat.diit.edu.ua:81/ft/index_ru.html
15. А.Е. Кононюк. Информациология. Общая теория информации. Кн.1. Киев: «Освіта України», 2011. 476 с.
ecat.diit.edu.ua:81/ft/index_ru.html
16. А.Е. Кононюк. Информациология. Общая теория информации. Кн.2. Киев: «Освіта України», 2011. 476 с.
ecat.diit.edu.ua:81/ft/index_ru.html
17. А.Е. Кононюк. Информациология. Общая теория информации. Кн.3. Киев: «Освіта України», 2011. 412 с.
ecat.diit.edu.ua:81/ft/index_ru.html
18. А.Е. Кононюк. Информациология. Общая теория информации. Кн.4. Киев: «Освіта України», 2011. 488 с.
ecat.diit.edu.ua:81/ft/index_ru.html

19. А.Е. Кононюк. Общая теория понятий. Кн.1. Киев: «Освіта України», 2014. 514с.

20. А.Е. Кононюк. Общая теория понятий. Кн.2. Киев: «Освіта України», 2014. 544с.

21. А.Е. Кононюк. Общая теория понятий. Кн.3. Киев: «Освіта України», 2014. 614с.

22. А.Е. Кононюк. Системология. Общая теория систем. Кн.1. Киев: «Освіта України», 2012. 564с. *ecat.diit.edu.ua:81/ft/index_ru.html*

23. А.Е. Кононюк. Системология. Общая теория систем. Кн.2. Ч.1. Киев: «Освіта України», 2014. 558с.
ecat.diit.edu.ua:81/ft/index_ru.html
24. А.Е. Кононюк. Системология. Общая теория систем. Кн.2. Ч.2. Киев: «Освіта України», 2014. 658с.
ecat.diit.edu.ua:81/ft/index_ru.html
25. А.Е. Кононюк. Системология. Общая теория систем. Кн.2. Ч.1. Киев: «Освіта України», 2014. 558с.
ecat.diit.edu.ua:81/ft/index_ru.html
26. А.Е. Кононюк. Общая теория распознавания. Кн.1. Киев: «Освіта України», 2012. 584 с. *ecat.diit.edu.ua:81/ft/index_ru.html*
27. А.Е. Кононюк. Общая теория распознавания. Кн.2. Киев: «Освіта України», 2012. 588 с. *ecat.diit.edu.ua:81/ft/index_ru.html*
28. А.Е. Кононюк. Консалтология. Общая теория консалтинга. Кн.1. Киев: «Освіта України», 2013. 448 с.
ecat.diit.edu.ua:81/ft/index_ru.html
29. А.Е. Кононюк. Консалтология. Общая теория консалтинга. Кн.2. Киев: «Освіта України», 2013. 412 с.
ecat.diit.edu.ua:81/ft/index_ru.html
30. А.Е. Кононюк. Консалтология. Общая теория консалтинга. Кн.3. Киев: «Освіта України», 2013. 520 с.
ecat.diit.edu.ua:81/ft/index_ru.html
31. А.Е. Кононюк. Консалтология. Общая теория консалтинга. Кн.4. Киев: «Освіта України», 2013. 508 с.
ecat.diit.edu.ua:81/ft/index_ru.html
32. А.Е. Кононюк. Дискретно-непрерывная математика. Начала. Кн.1. Киев: «Освіта України», 2012. 652с.
ecat.diit.edu.ua:81/ft/index_ru.html
33. А.Е. Кононюк. Дискретно-непрерывная математика. Множества. Кн.2. Ч.1. Киев: «Освіта України», 2012. 452с.
ecat.diit.edu.ua:81/ft/index_ru.html
34. А.Е. Кононюк. Дискретно-непрерывная математика. Множества. Кн.2. Ч.2. Киев: «Освіта України», 2013. 536 с.
ecat.diit.edu.ua:81/ft/index_ru.html
35. А.Е. Кононюк. Дискретно-непрерывная математика. Отношения. Кн.3. Ч. 1. Киев: «Освіта України», 2013. 552с.
ecat.diit.edu.ua:81/ft/index_ru.html

36. А.Е. Кононюк. Дискретно-непрерывная математика. Отношения. Кн.3. Ч. 2. Киев: «Освіта України», 2013. 548 с.
ecat.diit.edu.ua:81/ft/index_ru.html
37. А.Е. Кононюк. Дискретно-непрерывная математика. Алгебры. Кн.4. Ч.1. Киев: «Освіта України», 2011. 452с.
ecat.diit.edu.ua:81/ft/index_ru.html
38. А.Е. Кононюк. Дискретно-непрерывная математика. Алгебры. Кн.4. Ч.2. Киев: «Освіта України», 2011. 668 с.
ecat.diit.edu.ua:81/ft/index_ru.html

39. А.Е. Кононюк. Дискретно-непрерывная математика. Алгебры. Кн.4. Ч.3. Киев: «Освіта України», 2015. 488 с.
http://lib.sumdu.edu.ua/library/DocDescription?doc_id=640902

40. А.Е. Кононюк. Дискретно-непрерывная математика. Алгебры. Кн.4. Ч.4. Киев: «Освіта України», 2015. 548 с.

41. А.Е. Кононюк. Дискретно-непрерывная математика. Алгебры. Кн.4. Ч.5. Киев: «Освіта України», 2015. 528 с.

42. А.Е. Кононюк. Дискретно-непрерывная математика. Алгебры. Кн.4. Ч.6. Киев: «Освіта України», 2015. 608 с.

43. А.Е. Кононюк. Дискретно-непрерывная математика. Матрицы. Кн.5. Ч.1. Киев: «Освіта України», 2013. 612 с.
ecat.diit.edu.ua:81/ft/index_ru.html

44. А.Е. Кононюк. Дискретно-непрерывная математика. Матрицы. Кн.5. Ч.2. Киев: «Освіта України», 2013. 500 с.
ecat.diit.edu.ua:81/ft/index_ru.html

45. А.Е. Кононюк. Дискретно-непрерывная математика. Матрицы. Кн.5. Ч.3. Киев: «Освіта України», 2013. 520 с.
ecat.diit.edu.ua:81/ft/index_ru.html
46. А.Е. Кононюк. Дискретно-непрерывная математика. Матрицы. Кн.5. Ч.4. Киев: «Освіта України», 2013. 508 с.
ecat.diit.edu.ua:81/ft/index_ru.html

47. А.Е. Кононюк. Дискретно-непрерывная математика. Матрицы. Кн.5. Ч.5. Киев: «Освіта України», 2013. 672 с.
ecat.diit.edu.ua:81/ft/index_ru.html
48. А.Е. Кононюк. Дискретно-непрерывная математика. Поверхности. Кн.6. Ч.1. Киев: «Освіта України», 2012. 652с.
ecat.diit.edu.ua:81/ft/index_ru.html
49. А.Е. Кононюк. Дискретно-непрерывная математика. Графы. Кн.7. Ч.1 Киев: «Освіта України», 2014. 652с.
ecat.diit.edu.ua:81/ft/index_ru.html
50. А.Е. Кононюк. Дискретно-непрерывная математика. Графы. Кн.7. Ч.2 Киев: «Освіта України», 2014. 552с.
ecat.diit.edu.ua:81/ft/index_ru.html
51. А.Е. Кононюк. Дискретно-непрерывная математика. Графы. Кн.7. Ч.3 Киев: «Освіта України», 2015. 512с.
ecat.diit.edu.ua:81/ft/index_ru.html
52. А.Е. Кононюк. Дискретно-непрерывная математика. Графы. Кн.7. Ч.4 Киев: «Освіта України», 2015. 552с.
ecat.diit.edu.ua:81/ft/index_ru.html

53. А.Е. Кононюк. Дискретно-непрерывная математика. Графы. Кн.7. Ч.5 Киев: «Освіта України», 2015. 660с.

54. А.Е. Кононюк. Обобщенная теория моделирования. Кн.1. Ч.1 Киев: «Освіта України», 2012. 602с.
ecat.diit.edu.ua:81/ft/index_ru.html
55. А.Е. Кононюк. Обобщенная теория моделирования. Кн.1. Ч.2 Киев: «Освіта України», 2012. 708с. *ecat.diit.edu.ua:81/ft/index_ru.html*
56. А.Е. Кононюк. Обобщенная теория моделирования. Кн.1. Ч.3 Киев: «Освіта України», 2012. 568с. *ecat.diit.edu.ua:81/ft/index_ru.html*
57. А.Е. Кононюк. Обобщенная теория моделирования. Кн.2. Киев: «Освіта України», 2012. 548с. *ecat.diit.edu.ua:81/ft/index_ru.html*
58. А.Е. Кононюк. Обобщенная теория моделирования. Кн.3. Ч.1 Киев: «Освіта України», 2012. 636с. *ecat.diit.edu.ua:81/ft/index_ru.html*
59. А.Е. Кононюк. Обобщенная теория моделирования. Кн.3. Ч.2 Киев: «Освіта України», 2012. 448с. *ecat.diit.edu.ua:81/ft/index_ru.html*
60. А.Е. Кононюк. Обобщенная теория моделирования. Кн.3. Ч.3 Киев: «Освіта України», 2013. 588с. *ecat.diit.edu.ua:81/ft/index_ru.html*
61. А.Е. Кононюк. Основы теории оптимизации. Кн.1. Киев: «Освіта України», 2011. 602с. *ecat.diit.edu.ua:81/ft/index_ru.html*

62. А.Е. Кононюк. Основы теории оптимизации. Кн.2. Ч.1. Киев: «Освіта України», 2011. 552с. *ecat.diit.edu.ua:81/ft/index_ru.html*
63. А.Е. Кононюк. Основы теории оптимизации. Кн.2. Ч.2. Киев: «Освіта України», 2011. 616с. *ecat.diit.edu.ua:81/ft/index_ru.html*
64. А.Е. Кононюк. Основы теории оптимизации. Кн.2. Ч.3. Киев: «Освіта України», 2012. 456с. *ecat.diit.edu.ua:81/ft/index_ru.html*
65. А.Е. Кононюк. Основы теории оптимизации. Кн.2. Ч.4. Киев: «Освіта України», 2012. 512с. *ecat.diit.edu.ua:81/ft/index_ru.html*
66. А.Е. Кононюк. Основы научных исследований. Кн.1. Киев: «Освіта України», 2011. 508с. *ecat.diit.edu.ua:81/ft/index_ru.html*
67. А.Е. Кононюк. Основы научных исследований. Кн.2. Киев: «Освіта України», 2011. 452с. *ecat.diit.edu.ua:81/ft/index_ru.html*
68. А.Е. Кононюк. Основы научных исследований. Кн.3. Киев: «Освіта України», 2011. 456с. *ecat.diit.edu.ua:81/ft/index_ru.html*
69. А.Е. Кононюк. Основы научных исследований. Кн.4. Киев: «Освіта України», 2011. 456с. *ecat.diit.edu.ua:81/ft/index_ru.html*

70. А.Е. Кононюк. Общая теория коммуникаций. Кн.1. Киев: «Освіта України», 2014. 488с.

71. А.Е. Кононюк. Нейроні мережі і генетичні алгоритми. Киев: «Корнійчук», 2010. 448с. *ecat.diit.edu.ua:81/ft/index_ru.html*

72. Кононюк А. Е. Обобщенная теория познания и созидания. [В 2 кн.] Кн. 1 : Начала / А. Е. Кононюк. — Киев : Освіта України, 2013. *ecat.diit.edu.ua:81/ft/index_ru.html*

73. Кононюк А. Е. Обобщенная теория познания и созидания. [В 2 кн.] Кн. 2 : Теория познания. Ч. 1 / А. Е. Кононюк. — Киев : Освіта України, 2013 *ecat.diit.edu.ua:81/ft/index_ru.html*

74. Кононюк А. Ю. Вища математика. (Модульна технологія навчання) : навчальний посібник : в 2 кн. / А. Ю. Кононюк. — Київ : КНТ, 2009 — Кн. 1. — 2009. — 702 с. *ecat.diit.edu.ua:81/ft/index_ru.html*

75. Кононюк А. Ю. Вища математика. (Модульна технологія навчання) : навчальний посібник : в 2 кн. / А. Ю. Кононюк. — Київ : КНТ, 2009 Кн. 2. — 2009. — 790 с. *ecat.diit.edu.ua:81/ft/index_ru.html*

76. А.Е. Кононюк. Дискретно-непрерывная математика. Поверхности. Кн.6. Ч.2. Киев: «Освіта України», 2012. 652с. http://www.dut.edu.ua/ua/lib/127/category/96/view/1297

77. А.Е. Кононюк. Дискретно-непрерывная математика. Пространства. Кн.8. Ч.1. Киев: «Освіта України», 2016. 748 с. http://www.dut.edu.ua/ua/lib/1/category/96/view/1439

78. А.Е. Кононюк. Дискретно-непрерывная математика. Пространства. Кн.8. Ч.2. Киев: «Освіта України», 2016. 480с. http://lib.sumdu.edu.ua/library/DocDescription?doc_id=640775

79. А.Е. Кононюк. Истины и информация (фундаментальная теория представления истин и информации). К.1. Киев: «Освіта України», 2016. 568с.
80. А.Е. Кононюк. Истины и информация (фундаментальная теория представления истин и информации). К.2. Киев: «Освіта України», 2016. 558с.

81. А.Е. Кононюк. Истины и информация (фундаментальная теория представления истин и информации). К.3. Киев: «Освіта України», 2016. 588с.
82. А.Е. Кононюк. Истины и информация (фундаментальная теория представления истин и информации). К.4. Киев: «Освіта України», 2016. 552с
83. А.Е. Кононюк. Истины и информация (фундаментальная теория представления истин и информации). К.5. Киев: «Освіта України», 2016. 836 с
84. А.Е. Кононюк. Истины и информация (фундаментальная теория представления истин и информации). К.6. Киев: «Освіта України», 2016. 576 с

85. Алберг Дж., Нильсон Э., Уолш Дж. Теория сплайнов и ее приложения.—М.: Мир, 1972 (Ahlberg J.H., Nilson E.N., Walsh J.L. The

Theory of Splines and their Applications.—New York: Academic Press, 1967).

86. Березин И.С., Жидков Н.П. Методы вычислений. Т. 1,2.—М.: Наука, 1962, 1966.

87. Гардан П., Люка М. Машинная графика и автоматизация конструирования.— М.: Мир, 1987 (Techniques Graphiques Interactives et C.A.O./par Michel Lucas et Yvon Gar dan.—Prance: Hermes Publishing, 1983).

88. Гильберт Д., Кон-Фоссен С. Наглядная геометрия.—М.: Наука, 1981 (Hilbert D., Cohn-Vossen S. Anschauliche Geometrie.—Berlin: 1932).

89. Де Бор К. Практическое руководство по сплайнам.—М.: Радио и связь, 1985 (De Boor C. A Practical Guide to Splines.—Berlin: Springer, 1879).

90. Демидович Б.П., Марон И.А. Основы вычислительной математики.—М.: Наука, 1970.

91. Дубровин Б.А., Новиков С.П., Фоменко А.Т. Современная геометрия.— М.: Наука, 1986.

92. Ильин В. А., Позняк Э.Г. Аналитическая геометрия.—М.: Наука, 1981.

93. Ильин В. А., Позняк Э.Г. Линейная алгебра.—М.: Наука, 1984.

94. Калиткин Н.Н. Численные методы.—М.: Наука, 1978.

95. Ласло М. Вычислительная геометрия и компьютерная графика на С++ К—М.: Бином, 1997 (Laszlo M. J. Computational Geometry and Computer Graphics in C++.—Prentice Hall, 1996).

96. Препарата Ф., Шеймос М. Вычислительная геометрия.—М.: Мир, 1989 (Preparata F. P., Sham os M. Computational Geometry: An Introduction.— New York, Berlin, Tokyo: Springer-Verlag, 1985).

97. Роджерс Д., Адамс Дж. Математические основы машинной графики.—М.: Машиностроение, 1980 (Rogers D.F., Adams J. A. Mathematical Elements for Computer Graphics.—McGrow-Hill, 1976).

98. А.Е. Кононюк. Дискретно-непрерывная математика. Математическая логика. Кн.9. Ч.1 Киев: «Освіта України», 2017. 464с.
99. А.Е. Кононюк. Дискретно-непрерывная математика. Математическая логика. Кн.9. Ч.2 Киев: «Освіта України», 2017. 564с.
100. А.Е. Кононюк. Дискретно-непрерывная математика. Математическая логика. Кн.9. Ч.3 Киев: «Освіта України», 2017. 424с.
101. А.Е. Кононюк. Основы фундаментальной теории искусственного интеллекта. Кн. 1. Киев: «Освіта України», 2017. 730с
102. А.Е. Кононюк. Основы фундаментальной теории искусственного интеллекта. Кн. 2. Киев: «Освіта України», 2017. 660с

103. Яншин В. В., Калинин Г. А. Обработка изображений на языке Си для IBM PC: Алгоритмы и программы. — М.: Мир, 1994. — 240 с.

104. Роджерс Д. Алгоритмические основы машинной графики: Пер. с англ. - М.: Мир, 1989. – 512 с.

105. Роджерс Д., Адамс Дж. Математические основы машинной графики: Пер. с англ. – М.: Мир, 2001. – 604 с.

106. Тихомиров Ю. Программирование трехмерной графики. – СПб: BHV – Санкт-Петербург, 1998. – 256 с.

107. Фоли Дж., вэн Дэм А. Основы интерактивной машинной графики: В 2-х кн., Кн. 1. / Пер. с англ. – М.: Мир, 1985 – 368 с.

108. Фоли Дж., вэн Дэм А. Основы интерактивной машинной графики: В 2-х кн., Кн. 2. / Пер. с англ. – М.: Мир, 1985 – 368 с.

109. Шикин Е. В., Боресков А. В. Зайцев А. А. Начала компьютерной графики. – М.: ДИАЛОГ-МИФИ, 1993. – 138 с.

110. Шикин Е. В., Боресков А. В. Компьютерная графика. Полигональные модели. – М.: ДИАЛОГ-МИФИ, 2000. – 464 с.

Рекомендуемые Интернет ресурсы

- Компьютерная графика на кафедре ОСУ
- Лекции по компьютерной графике
- Библиотека различных алгоритмов, в том числе и по компьютерной графике
- Курс лекций Московского государственного университета
- Введение в компьютерную графику. Курс ВМиК МГУ
- Курс компьютерной графики Новосибирского Государстенного Техническгого Университета (НГТУ)

Ссылки по Macromedia Flash

- Изучаем Flash
- Клуб "Флэшеров"
- FlashKit

Ссылки по OpenGL и DirectX

- Центр информационных технологий. (Томское зеркало)
- сайт Игоря Тарасова, автора книги "Основы OpenGL"
- OpenGL solution for Delphi

Ссылки по Adobe PhotoShop и CorelDraw

- Изучаем Photoshop (Томский сайт)
- Рецепты для Photoshop (Томский сайт)
- Уроки Photoshop (Томский сайт)
- PhotoShop в Томске
- Анатомия Adobe PhotoShop (www.psd.ru)
- Photoshop tutorials
- Различные эффекты в Photoshop
- Школа Photoshop
- Несколько статей о PhotoShop
- www.corel.ru

животного, можно искать ключ к биологическому распоз-□□
□