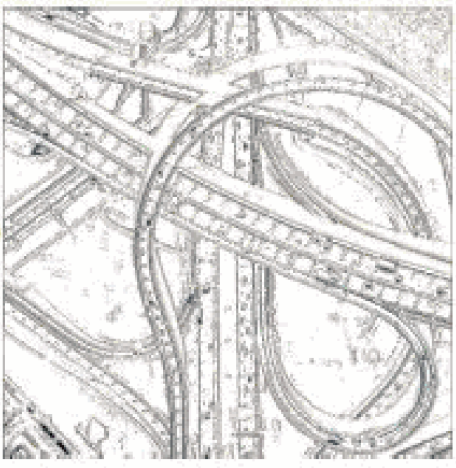
А. Ю. Гришенцев
А. Г. Коробейников

МЕТОДЫ И МОДЕЛИ ЦИФРОВОЙ ОБРАБОТКИ ИЗОБРАЖЕНИЙ

$$F : X \to Y$$
$$F^{-1} : Y \to X$$



А. Ю. Гришенцев А. Г. Коробейников

МЕТОДЫ И МОДЕЛИ ЦИФРОВОЙ ОБРАБОТКИ ИЗОБРАЖЕНИЙ

Санкт-Петербург
Издательство Политехнического университета
2014

УДК 004.92; 004.932; 004.05
ББК 32.973.26-04; 32.973.26-018.2
Г85

Р е ц е н з е н т ы:
Доктор физико-математических наук, профессор,
директор Санкт-Петербургского филиала Федерального государственного
бюджетного учреждения науки Института земного магнетизма,
ионосферы и распространения радиоволн им. Н. В. Пушкова
Российской академии наук *Ю. А. Копытенко*
Доктор технических наук, профессор, декан факультета
информатики и вычислительной техники Федерального государственного
бюджетного образовательного учреждения высшего профессионального
образования «Поволжский государственный технологический
университет» *И. Г. Сидоркина*

Гришенцев А. Ю. **Методы и модели цифровой обработки изображений** / А. Ю. Гришенцев, А. Г. Коробейников. – СПб. : Изд-во Политехн. ун-та, 2014. – 190 с.

Монография посвящена некоторым методам и моделям обработки функциональных изображений, научным основам построения средств компьютерной графики, методов геометрического моделирования проектируемых объектов и синтеза виртуальной реальности. Наряду с фундаментальными методами рассмотрены авторские, нашедшие успешное применение в практике. В работе приведено значительное количество практических примеров, реализованных на языке C++ и в среде MatLab. Книга может быть интересна студентам, аспирантам и инженерам, специализирующимся в области цифровой обработки.

Редактор *Л. Г. Позднякова*
Компьютерная верстка *М. В. Герасимова*

ISBN 978-5-7422-4892-7

СОДЕРЖАНИЕ

ПРЕДИСЛОВИЕ

Присущая человеку жажда знания является одной из движущих сил развития цивилизации.

В наше время информационные технологии получили особое значение. С конца XX–начала XXI вв. благодаря повсеместному распространению электронных вычислительных средств (ЭВС) произошли значительные изменения на самых различных уровнях организации и управления ресурсами, в том числе информационными. Широкие круги общества получили доступ практически к любой информации. Исследователи имеют возможность решать задачи, требующие одновременной работы множества мощнейших вычислительных систем. Стимулированные потребительским спросом значительно трансформировались и развились области знаний, связанные с информационными технологиями. Появились свежие инженерные решения, сформировались новые научные направления. Тенденции развития современных вычислительных устройств предполагают применение таких технологий, как параллельные и облачные вычисления. Доступность существенных объёмов памяти и рост тактовой частоты процессоров позволяют решать практические задачи с помощью методов, которые ранее вследствие вычислительной затратности не могли быть решены на существующих вычислительных платформах. Благодаря современным решениям, например GPGPU (General-Purpose Graphics Processing Units), реализующим неграфические вычисления на графических процессорах, обыкновенный персональный компьютер способен оперативно обрабатывать ресурсоемкие задачи.

Цифровые сигналы в большинстве случаев являются результатом преобразования информации, получаемой от аналоговых датчиков, а в некоторых случаях полностью производятся (синтезируются) цифровыми устройствами. Цифровая обработка сигналов (ЦОС), по сравнению с аналоговой, во многих случаях позволяет существенно поднять качественный уровень передачи, хранения и преобразования данных.

Повышение многообразия цифровых устройств и увеличение потока информации, требующей обработки, передачи и хранения, определяют актуальность исследований и анализа методов цифровой обработки сигналов и, в частности, цифровых изображений. Существенная часть инженерных, научных задач решается методами обработки изображений [4, 81, 125]. Возможность визуализации процесса и результатов обработки данных, т.е. синтез цифровых графических изображений, является стандартным функционалом современных приложений [101, 159, 164, 166].

В данной монографии рассмотрены некоторые элементы теории ЦОС, изучены аспекты нового научно-практического направления формализации и обработки цифровых сигналов, в первую очередь графических изображений, как полевых структур.

Практическая ценность монографии заключается в анализе некоторых современных принципов ЦОС, подробном рассмотрении алгоритмов и программ, реализованных на языке C++ и в программной среде MatLab.

Значительная часть исследований выполнена на базе Университета ИТМО (Санкт-Петербург), Санкт-Петербургского филиала Института земного магнетизма, ионосферы и распространения радиоволн им. Н. В. Пушкова РАН (ИЗМИРАН). Результаты исследований востребованы в проектах Национального научного центра «Никитский ботанический сад» (НБС, Крым), Санкт-Петербургского государственного политехнического университета (СПбГПУ), Государственного научного центра «Арктический и антарктический научно-исследовательский институт» (ААНИИ).

Монография рассчитана на специалистов в области цифровой обработки сигналов, она может быть полезна студентам и аспирантам.

Гришенцев А. Ю. выражает благодарность своим учителям Л. И. Иванову, Ю. П. Пирогову, К. Г. Короткову, А. Г. Коробейникову

1. Формат цифровых изображений

Формат цифровых изображений зачастую определяется архитектурой вычислительной машины, в первую очередь – организацией памяти и доступа к ней. Совместимость вычислительных систем подразумевает возможность работы с несколькими распространёнными форматами. Существует значительное число широко распространённых форматов, стандартных и являющихся практически (*de facto*, т.е. «на деле») стандартными, что, в свою очередь, требует множества их взаимных преобразований.

Для получения эффективных результатов при разработке программного обеспечения средствами ЭВС необходимо учитывать не только особенности форматов данных, но и восприятие их человеком.

1.1. Цифровые изображения

Термин «изображение» часто используют для обозначения художественного образа, например, литературное изображение, графическое изображение и т.д.

Изображение – объект, образ, явление, в той или иной степени подобное (но не идентичное) изображаемому или сам процесс их создания.

Изображением (*образом*) в математическом смысле называют результат функционального преобразования. *Преобразование* (*отображение*) множества X в множество Y обозначают $f : X \to Y$. Если для каждого $x \in X$ существует единственный элемент $y \in Y$, то $y = f(x)$. Для некоторого прообраза $Z \subseteq X$ множество $f(Z)$ есть изображение, или образ [6].

Говоря о *цифровых изображениях*, имеют в виду цифровые (дискретные) образы. В контексте этой работы *цифровым изображением* (или изображением) называются цифровые сигналы, полученные в результате некоторых отображений. В большинстве случаев рассматривается частный случай цифровых сигналов – *цифровые графические изображения*. Отметим, что цифровые сигналы в большинстве случаев являются образами реальных физических процессов; могут иметь только дискретные, в заданном диапазоне, значения; всегда имеют конечную область определения. Следует отметить, что цифровые графические изображения являются частным случаем *многомерных цифровых сигналов*.

В качестве объекта обработки и исследования в работе используются цифровые графические изображения, поскольку:

– они составляют наиболее значимый объём цифровой информации, накопленной человечеством;

– графические изображения удобны для восприятия человеком;

– несмотря на то что рассматриваемые методы применимы к самому широкому классу цифровых сигналов, изначально они создавались для обработки именно цифровых графических изображений.

Изначально превалирующее большинство сигналов существует в аналоговом виде. Зарегистрированные приёмниками, они могут быть преобразованы в цифровой вид с помощью аналого-цифрового преобразователя (АЦП). Критерием корректного преобразования аналогового сигнала в цифровой вид является возможность его восстановления с заданной точностью. Основополагающим законом, позволяющим производить такие преобразования, является опубликованная в 1933 г. *теорема Котельникова* [82], также называемая теоремой отсчётов, повторно «открытая» в 1949 г. Клодом Шенноном.

Теорема Котельникова гласит: любую функцию $x(t)$, состоящую из частот от нуля до f, можно передавать с любой точностью при помощи чисел, следующих друг за другом через $\frac{1}{2f}$ секунд, такая последовательность образует дискретные отсчёты с частотой дискретизации $2f$.

В 1999 г. Международный научный фонд Эдуарда Рейна (Германия) признал приоритет В. А. Котельникова[1], наградив его в номинации «фундаментальные исследования» премией «за впервые математически точно сформулированную и доказанную в аспекте коммуникационных технологий теорему отсчётов» [83].

1.2. Физиологические особенности восприятия информации

Во множестве практических задач цифровой обработки сигналов конечным объектом восприятия информации является человек. Например, с особенностями физиологического восприятия изображений связаны форматы, параметры и принципы графического отображения цифровых сигналов, в частности, цветовые схемы, частота обновления кадров, гамма-коррекция, разрешение оптико-электронных приборов и др.

В исследованиях Э. Вебера[2] показано, что определяющим связь между интенсивностью ощущения и силой раздражения воспринимается не абсолютный, а относительный прирост силы раздражителя (света, воздействующего на сетчатку глаза; звука, воспринимаемого органами слуха; груза, давящего на кожу):

---

[1] Владимир Александрович Котельников (24 августа, 1908, Казань – 11 февраля 2005, Москва) – инженер и учёный.

[2] Эрнст Генрих Вебер (24 июня 1795, Виттенберг – 26 января 1878, Лейпциг) – психофизиолог и анатом, брат физика Вильгельма Вебера.

$$\frac{\Delta P}{P} = \int \text{const}\,, \tag{1.1}$$

где $P, \Delta P$ – величина и приращение раздражителя. Минимальный прирост ощущения определяется как

$$\Delta S = c\frac{\Delta P}{P}\,, \tag{1.2}$$

где c – коэффициент пропорциональности. Проинтегрировав выражение (1.2) и полагая, что величина $S = 0$ (ощущение) при минимальном значении абсолютного порога $P_{\min}$, можно записать:

$$S = c\ln|P| - c\ln|P_{\min}| = c\ln\left|\frac{P}{P_{\min}}\right|.$$

Перейдя к десятичному основанию логарифма при помощи подстановки $k = \frac{c}{\lg e}$, получим закон Вебера [102]:

$$S = k\lg\left|\frac{R}{R_0}\right|. \tag{1.3}$$

Разнообразные раздражители принято классифицировать по *модальности* [102], различают световые, механические, химические, осмотические, электрические, тепловые и другие раздражители. Все раздражители подразделяют на адекватные и неадекватные по отношению к конкретной ткани, в частности, к рецепторному аппарату. Адекватным называют раздражитель, к восприятию воздействия которого ткань в ходе эволюции приспособилась. Адекватность раздражителя проявляется в том, что его пороговая величина значительно ниже, чем у неадекватных раздражителей.

Значительная разница в восприятии адекватных и неадекватных раздражителей свойственна *сенсорным системам* всех органов чувств [104]. Сенсорные системы образованы: рецепторами, реагирующими на сигналы внешней и внутренних сред организма; чувствительными нервными волокнами, проводящими сигналы от рецепторов в центральную нервную систему (ЦНС); структурами ЦНС, которые анализируют эти сигналы.

В соответствии с модальностью адекватного раздражителя, разнообразные рецепторные аппараты подразделяют на:

– фоторецепторы (обеспечивают восприятие света),

– механорецепторы (ответственны за восприятие механических раздражителей, в том числе виброакустических),

– хеморецепторы (получают информацию о химических свойствах окружающей среды). К хеморецепторам относятся обонятельные и вкусовые, а также некоторые рецепторы внутренних органов (интероцепторы),

– терморецепторы (воспринимают изменение температуры окружающей среды),

– осморецепторы (реагируют на изменение осмотического давления).

Нами рассмотрены далеко не все рецепторные аппараты человека, присутствующие в большом количестве во всех органах и тканях. По расположению рецепторных аппаратов различают экстерорецепторы, обращённые во внешнюю среду, и интерорецепторы, реагирующие на процессы во внутренней среде организма. У экстерорецепторов выражена специализация – высокая избирательная чувствительность к адекватным раздражителям. Из-за большой разницы в порогах адекватных и неадекватных раздражителей большинство экстерорецепторов принято считать приспособленными к восприятию стимулов одной модальности. Например, фоторецепторы глаза начинают воспринимать свет с минимальным потоком световой энергии порядка $10^{-17}-10^{-18}$ Вт, ощущение вспышки можно вызвать механическим воздействием на глаз (явление механического фосфена), для этого необходима энергия порядка 10^{-4} Вт, следовательно, разница между световым (адекватным) и механическим (неадекватным) раздражителем составляет порядка $10^{13}-10^{14}$ раз.

В. О. Самойлов отмечает, что чувствительность фоторецепторов человека необычайно высока [104]. Возбуждение палочки происходит при поглощении одного кванта света, человек способен ощущать такой свет, который приводит к возбуждению не менее 5–10 палочек, поглощающих соответствующее число фотонов. Пороги чувствительности одного порядка были получены различными исследователями. С. И. Вавилов[3] установил порог в 9 фотонов (к примеру, С. Гехт – 6 фотонов, Н. И. Пинегин – 2 фотона), такие результаты исследования позволили ему сделать вывод о том, что зрительный анализатор человека близок по своим характеристикам к идеальному прибору. Диапазон воспринимаемой яркости человеком $10^{-6}-10^{6}$ кд·м$^{-2}$, что соответствует 12 порядкам, а по некоторым оценкам – 14.

Фоторецепторы сосредоточены на сетчатке глаза, каждый глаз человека содержит примерно $1.25\cdot10^{8}$ палочек и $5\cdot10^{6}$ колбочек. Фактически фоторецепторная система человека является дискретной. Зрительный нерв содержит порядка 10^{6} волокон, по ним в мозг направляются сигналы от $130\cdot10^{6}$ фоторецепторов. Один аксон ганглиозной клетки в среднем соби-

[3]Сергей Иванович Вавилов (12 марта 1891, Москва – 25 января 1951, Москва) – советский физик, основатель научной школы физической оптики в СССР, академик (1932) и президент АН СССР (с 1945 года). Лауреат четырёх Сталинских премий.

рает информацию от множества рецепторных клеток. Все фоторецепторы, посылающие сигналы к этой ганглиозной клетке, составляют её рецепторное поле. Число рецепторных полей, образованных палочками, гораздо больше числа колбочковых полей. Каждая колбочка, находящаяся в центральной ямке жёлтого пятна сетчатки, связана с одной ганглиозной клеткой, что определяет наилучшую разрешающую способность центрального колбочкового зрения. Ганглиозные клетки, к которым сигналы приходят от палочек, получают сигналы от многих тысяч палочек. Поэтому разрешающая способность палочкового зрения существенно ниже колбочкового, при этом первое обеспечивает максимальную чувствительность. Палочковое зрение называют *скотопическим*, световая чувствительность скотопического зрения, по ощущениям зрительного восприятия, примерно на три порядка выше колбочкового (*фотопического*).

Первичный механизм возбуждения палочек светом связан со сложными превращениями высокомолекулярного соединения (родопсина) в фоторецепторной мембране. Максимумы спектра поглощения родопсина приходятся на длину волны λ=498 (видимое излучение) и 278 нм (ультрафиолетовое, невидимое, излучение). Поглощение в видимом диапазоне приводит к возбуждению родопсина и последующему восприятию света, ультрафиолетовое (УФ) излучение не может возбудить родопсин, поскольку оно практически полностью поглощается преломляющими средами глаза. Спектр поглощения родопсина хорошо «приспособлен» к восприятию достаточно широкой области солнечного излучения, но лучше всего этот зрительный пигмент поглощает в зелёной части спектра (рис. 1.1, D – оптическая плотность родопсина). В процессе эволюции организм приобрёл зрительные пигменты, эффективно воспринимающие ту область спектра, которой принадлежит значительная доля отфильтрованного атмосферой излучения абсолютно чёрного тела при T=6000 К, что близко к температуре поверхности Солнца.

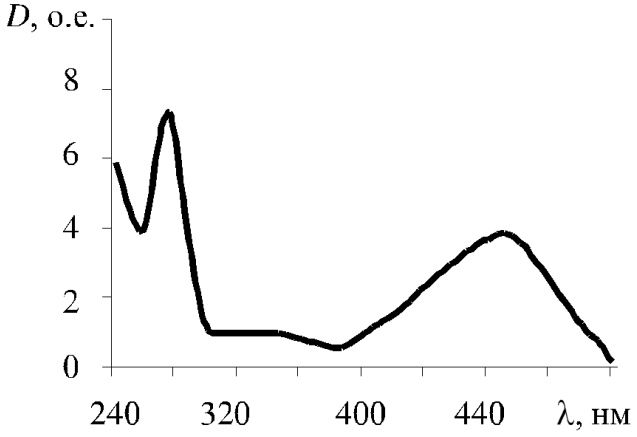


Рис. 1.1. Спектр поглощения родопсина

С функционированием колбочек, сосредоточенных в центральной ямке сетчатки, связана предельная острота зрения человека. Благодаря

существованию колбочковых пигментов трёх типов человек обладает цветным зрением. В результате у человека и приматов присутствуют колбочки трёх типов, условно названные «красными», «зелёными», «синими», каждая из которых характеризуется содержанием особого пигмента. Пигмент «красных» колбочек имеет фиолетовый цвет и потому называется йодопсином (от гр. ιωδωσ – фиолетовый), «красные» колбочки способны хорошо поглощать свет в жёлто-красной области солнечного спектра (λ≈562 нм). Максимум поглощения пигмента «зелёных» колбочек приходится на λ≈500 нм, пигмент «синих» колбочек лучше всего поглощает фиолетовые и синие лучи с максимумом поглощения на λ≈449 нм. Спектры поглощения колбочковых пигментов приведены на рис. 1.2 (*p* – относительное поглощение излучения; *1* – «синие», *2* – «зеленые», *3* – «красные»).

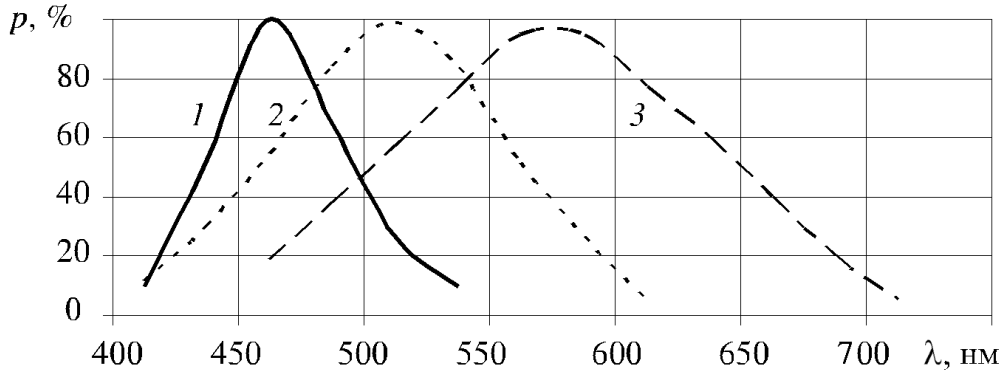


Рис. 1.2. Спектр поглощения колбочек

Строение органов зрения и слуха человека чрезвычайно сложно, им свойственны различные способы передачи и преобразования сигналов как в цифровом, так и в аналоговом виде. Например, пространственная структура расположения фоторецепторов на сетчатке дискретна, в результате проекции изображения фотосенсорная система в каждый момент времени способна формировать образ воспринимаемой информации с некоторым конечным разрешением (возбуждение импульсов в нейронной сети также обладает свойствами дискретности).

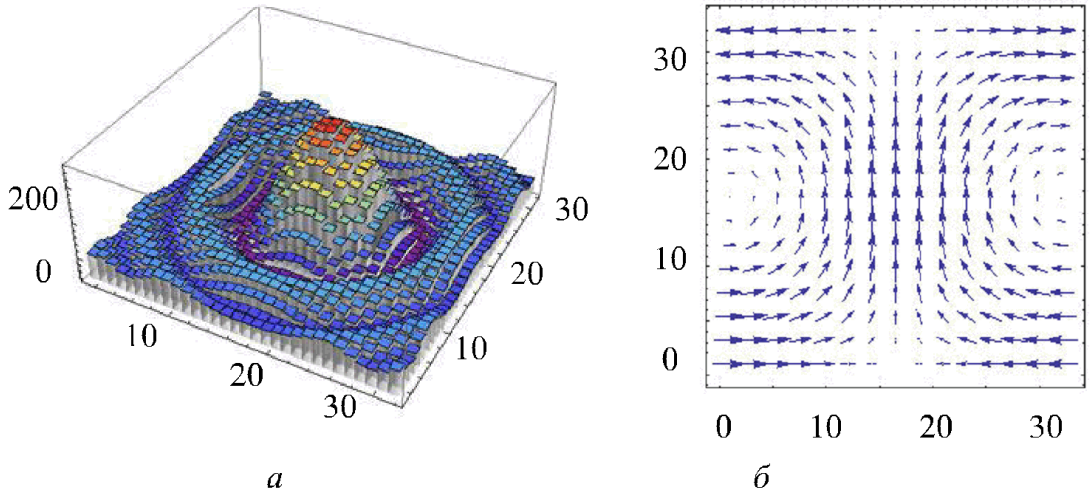
1.3. Тензорное поле – формат представления многомерных цифровых сигналов

Все физические процессы происходят в четырёхмерном пространственно-временном континууме – окружающий нас мир имеет три пространственные координаты и одну временную. В некоторых случаях могут возникать задачи обработки данных (сигналов), рассматриваемых в пространствах, имеющих более четырёх измерений.

Формализация способа записи многомерных сигналов требует особого внимания. Каждый элемент многомерного сигнала может быть представлен множеством значений, причём мощность этого множества может не

соответствовать размерности пространства, в котором рассматривается цифровой сигнал. Элементы цифрового сигнала могут быть представлены скалярными и векторными величинами, а сигналы могут быть рассмотрены в различных координатных пространствах. Такое многообразие представлений цифровых сигналов требует определённой систематизации. Возможность несовпадения мерности пространства, в котором рассматривается сигнал, и мерности пространства значений сигнала, не позволяет априори считать сигналы векторными или скалярными полями.

Для формального описания цифровых сигналов наиболее подходит тензорное поле (рис. 1.3, *а* – тензор нулевого ранга, скаляр, *б* – тензор первого ранга, вектор в двумерном пространстве–плоскости), когда каждому элементу, характеризуемому набором координат в пространстве сигнала, ставится в соответствие тензор одного типа. Такой подход позволит обобщить множество действий на базе определённых операций над тензорами.



а *б*

Рис. 1.3. Двумерные дискретные тензорные поля

Будем называть *многомерной* систему, объекты которой исследуются в конечномерном пространстве, допускающем размерность $n \geq 2$.

Для обозначения размерности пространства, в котором рассматриваются сигналы, будем использовать упорядоченное ортонормированное множество $R^n = \langle r_0, r_2, ..., r_{n-1} \rangle$. Запись $f(R^n)$ означает, что сигнал рассматривается в n-мерном пространстве. Цифровые сигналы обозначим следующим образом: $f[r_0, r_2, ..., r_{n-1}]$, или $f[R^n]$. Очевидно, что одномерные сигналы могут рассматриваться как частный случай. Значению сигнала в каждой точке пространства ставится в соответствие тензор X типа (p, q), т.е. p раз *ковариантный* и q раз *контравариантный*. Обозначать тензоры будем $x_{i_1 i_2 ... i_p}^{j_1 j_2 ... j_q}$, индексы $i_1 i_2 ... i_q$ называют ковариантными (соответствуют ковариантным координатам $x_{i_1 i_2 ... i_p}$ в каждом базисе $\mathbf{e}^i$), $j_1 j_2 ... j_q$ – контра-

вариантными (соответствуют контравариантным координатам $x^{j_1 j_2 \dots j_q}$ в каждом базисе $\mathbf{e}_j$). Сумму (p+q) называют рангом, или валентностью, тензора. Тензор будем рассматривать в произвольном (необязательно евклидовом) вещественном k -мерном линейном пространстве L^k.

Частным случаем тензора нулевого ранга является скаляр x. Тензору первого ранга (с единичной валентностью) соответствует ковариантный вектор x_i или контравариантный x^j. В большинстве случаев для записи цифровых сигналов достаточно скаляров и векторов (см. рис. 1.3).

В формализации тензорного исчисления используются краткие записи групповых операций с однотипными слагаемыми – соглашения. Представим выражение, составленное из сомножителей, которые снабжены конечным числом индексов – нижних и верхних. При этом все индексы обозначены различными символами. Если в выражении два одинаковых индекса: один верхний, а другой нижний, то по этим индексам выполняется суммирование. Например, для выражения

$$y_i = \sum_i b_{im} x^m$$

краткая форма записи будет иметь вид:

$$y_i = b_{im} x^m , \qquad (1.4)$$

где y_i – координаты ковариантного вектора первого ранга; x^m – координаты контравариантного вектора первого ранга; b_{im} – координаты ковариантного вектора второго ранга (матрицы).

Мы не будем рассматривать все формальные определения и операции тензорного исчисления [22, 67], поскольку это выходит за рамки работы.

Большинство физических законов можно выразить дифференциальными соотношениями между физическими величинами, являющимися тензорными полями [94]. Цифровые сигналы соответствуют образам некоторых реальных физических процессов или виртуальных моделей, по этой причине тензорные исчисления являются удобным способом обработки цифровых сигналов, соответствующих отображению реальных физических.

Применение тензорного исчисления к многомерным цифровым сигналам позволяет формализовать некоторые особенности сигналов, а операции над тензорами – расширить операционное пространство, доступное для действий с ними.

1.4. О цветовых пространствах

Зрительная система человека воспринимает информацию покомпонентно, любой цвет может быть получен в результате синтеза, т.е. комбинаций – суммы или разности компонентов различной интенсивности. Обычно используют три компонента – *базовые цвета*: красный, зелёный и синий, или RGB (Red, Green, Blue). С точки зрения формирования цветопередачи цветовые модели можно разделить на аддитивные, разностные и смешанные. Например, модель RGB является аддитивной; CMYK – разностной; а YCbCr – смешанной, в которой компонент Y – аддитивный, Cb и Cr – разностные.

В настоящее время имеется множество моделей цветовых пространств и их вариаций, а соответственно – средств обработки и воспроизведения цветных изображений. На базе цветовых моделей формируют цветовое пространство, которое можно рассматривать как геометрическое, в котором задана система координат, связанная с цветовой моделью. Наиболее распространены трёхкомпонентные цветовые модели и пространства. В практике ЦОС для представления цветовых пространств наиболее широко используется прямоугольная, реже – цилиндрическая система координат. В цифровых сигналах число возможных значений каждого компонента определяет совокупное возможное количество цветовых оттенков, например, при восьмиразрядном представлении каждого компонента в трёхмерном цветовом пространстве – 256, общее число цветов $256^3 = 16777216$, такого цветового разнообразия вполне достаточно для визуализации большинства цветных изображений.

Исследованиями и стандартизацией цветопередачи, в том числе в области цветоощущения, занимаются следующие организации:

– International Commission on Illumination, или CIE – Международная комиссия по освещению, официальный сайт: http://cie.co.at/;

– ISO, International Organization for Standartization – Международная организация стандартизации, официальный сайт: http://www.iso.org/;

– ITU, International Telecommunication Union – Международный союз электросвязи, официальный сайт: http://www.itu.int/;

– Всероссийский научно-исследовательский светотехнический институт им. С. И. Вавилова, официальный сайт: http://www.vnisi.ru/.

Эти организации разработали ряд стандартов и рекомендаций по классификации и использованию цветовых пространств.

Рассмотрим некоторые цветовые пространства и методы взаимного преобразования их моделей. Следует отметить, что широко используемые дискретные реализации моделей цветовых пространств не обеспечивают передачу всего спектра оттенков, воспринимаемых человеком (поскольку большинство моделей не охватывают спектра, кроме того, при дискретной

реализации непрерывный аналоговый спектр интерпретируется как линейчатый). Однако для решения большинства задач отображения графической информации вполне достаточно возможностей существующих моделей.

Линейное цветовое пространство CIE *xyY* предложено Международной комиссией по освещению, модель цветового пространства CIE *xyY* построена на основании зрительных особенностей восприятия «среднего наблюдателя». Цветовое пространство CIE *xyY* является трёхмерным, по оси *x* расположен цветовой градиент от синего до красного, по оси *y* – от синего до зелёного, по оси *Y* распределены значения интенсивности элементов цветовой плоскости *xy*. На рис. 1.4 (http://members.eunet.at/cie) выделена область, очерченная линиями красного и фиолетового, внутри нее находятся все видимые человеком цвета. Эта диаграмма получена в результате длительных исследований способностей цветоощущения человека. В центре диаграммы находится оцененная статистически для «среднего наблюдателя» точка равных интенсивностей цветовых компонентов, соответствующая белому цвету, а при движении по оси *Y* – различной интенсивности оттенка серого. На рис. 1.4 отображена графическая интерпретация соотношения CIE *xyY* с цветовыми моделями NTSC, RGB.

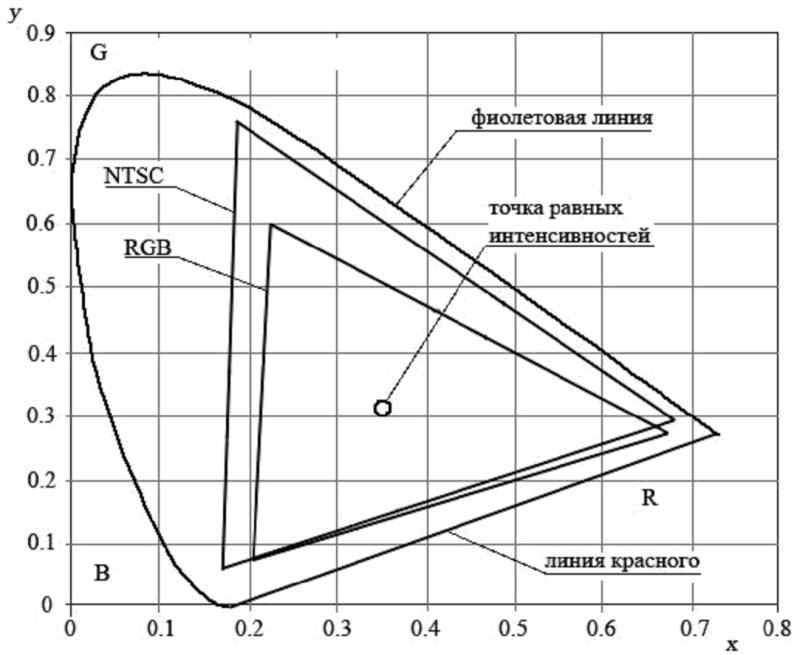


Рис. 1.4. Соотношение цветовых пространств RGB, NTSC на плоскости CIE xy

Все оттенки в модели RGB формируются путём сложения трёх компонентов с различной интенсивностью, поэтому RGB является аддитивной. Сочетание компонентов равной интенсивности даёт различные оттенки серого цвета: от чёрного до белого. RGB-модель построена с учётом трёхкомпонентного цветового восприятия человека и учитывает физиологические особенности, она наиболее широко используется в электронных уст-

ройствах визуализации цветной графической информации. Трёхкомпонентное цветовое пространство во многих случаях позволяет эффективно производить обработку и анализ графической информации, поэтому на работу с данными в цветовом пространстве RGB ориентированы многие вычислительные устройства и программное обеспечение.

CMYK-модель (Cyan – голубой, Magenta – пурпурный, Yellow – жёлтый и дополнительный цвет blacK – чёрный, или Key color) является подмножеством модели RGB и используется в основном для цветной триадной печати на белой поверхности. Модель CMYK, в отличие от RGB, является разностной, это означает, что получение требуемого оттенка основывается на вычитании из белого (цвета поверхности печати) соответствующей комбинации CMYK. Чёрный цвет необходим для возможности изменения яркостной составляющей и получения некоторых оттенков. Число доступных цветовых оттенков для модели CMYK существенно меньше, чем, например, для RGB. Отображение цвета в пространстве CMYK существенно зависит от яркости и белизны поверхности печати.

Трёхмерная базовая модель YUV применяется в аналоговом телевизионном вещании. Будучи производной от RGB-модели YUV содержит три компонента: Y определяет общий уровень яркости, аддитивный компонент, а U и V – цветоразностные. Яркость рассчитывается путём взвешивания интенсивностей красного, зелёного и синего, разностные компоненты формируются путём вычитания яркости из синего и красного. Обратную совместимость форматов цветного и черно-белого телевизионных сигналов обеспечивает яркостная составляющая, обрабатывать которую возможно вне зависимости от цвета: например, производить изменение динамического диапазона, в частности, гамма-коррекцию. Цветовое пространство, построенное на базе YUV-модели (при диапазонах значений Y = 0–1, U = –0.6–0.6 и V = –0.6–0.6), содержит значительно больше цветовых оттенков, чем RGB при допустимых значениях для всех компонентов 0–1. Компоненты цветового пространства, прошедшие гамма-коррекцию, будем обозначать как $Y'U'V'$, взаимное преобразование R',G',B' в $Y'U'V'$ и обратно можно выполнять с помощью следующих выражений:

$$\begin{cases} Y' = 0.299R' + 0.587G' + 0.114B', \\ U' = -0.147R' - 0.289G' + 0.436B' = 0.429(B' - Y'), \\ V' = 0.615R' - 0.515G' - 0.100B' = 0.877(R' - Y') \end{cases} \qquad (1.5)$$

и

$$\begin{cases} R' = Y' + 1.140V', \\ G' = Y' - 0.394U' - 0.581V', \\ B' = Y' + 2.032U'. \end{cases} \qquad (1.6)$$

YCbCr-модель цветового пространства, используемая для цифрового видео, разработана в рамках рекомендации ITU-R BT.601-7 от 03/2011 [153], YCbCr-модель построена на базе YUV-модели. Компоненты цветового пространства, прошедшие гамма-коррекцию, будем обозначать как $Y'Cb'Cr'$, взаимное преобразование R',G',B' в $Y'Cb'Cr'$ и обратно можно производить с помощью выражений [155]:

$$\begin{cases} Y' = 0.257R' + 0.504G' + 0.098B' + 16, \\ Cb' = -0.148R' - 0.291G' + 0.439B' + 128, \\ Cr' = 0.439R' - 0.368G' - 0.071B' + 128 \end{cases} \quad (1.7)$$

и

$$\begin{cases} R' = 1.164(Y' - 16) + 1.596(Cr' - 128), \\ G' = 1.164(Y' - 16) - 0.813(Cr' - 128) - 0.392(Cb' - 128), \\ B' = 1.164(Y' - 16) + 2.017(Cb' - 128). \end{cases} \quad (1.8)$$

Для преобразования значений, описывающих цвет без гамма-коррекции, используемого, например, в цифровом телевидении, в пространствах YCbCr и RGB, применяют следующие выражения:

$$\begin{cases} Y = 0.299R + 0.587G + 0.114B, \\ Cb = -0.16874R - 0.33126G + 0.5B + 128, \\ Cr = 0.5R - 0.41869G - 0.08131B + 128 \end{cases} \quad (1.9)$$

и

$$\begin{cases} R = Y + 1.402Cr - 179.456, \\ G = Y - 0.34414Cb - 0.71414Cr + 135.45984, \\ B = Y + 1.772Cb - 226.816. \end{cases} \quad (1.10)$$

Вариация Y_{709}CbCr-модели, распространённая в компьютерных системах, цифровом телевидении, также применяется при компрессии графических изображений:

$$\begin{cases} Y_{709} = 0.183R + 0.614G + 0.062B + 16, \\ Cb = -0.101R - 0.338G + 0.439B + 128, \\ Cr = 0.439R - 0.399G - 0.040B + 128 \end{cases} \quad (1.11)$$

и

$$\begin{cases} R = 1.16414Y_{709} + 1.79237Cb - 0.00109824Cr - 247.909, \\ G = 1.16414Y_{709} - 0.534308Cb - 0.213096Cr + 77.0414, \\ B = 1.16414Y_{709} + 0.000987185Cb + 2.11358Cr - 289.291. \end{cases} \quad (1.12)$$

YCbCrK-модель специфична для JPEG-сжатия изображений, это расширение YCbCr-модели, содержащее дополнительный канал черного (blacK) цвета. JPEG-сжатие происходит эффективнее, если информация о цвете и информация о яркости изображения преобразуются независимо друг от друга.

PhotoYCC-модель, разработанная на базе рекомендаций ITU601 и ITU709 [153, 154], применяется для кодирования цветных фотографий. Модель содержит яркостную составляющую (Y) и две цветоразностные $C1$ и $C2$. Отображаемый спектр цветовой модели PhotoYCC-модель несколько шире спектра, воспроизводимого большинством средств визуализации графической информации. При разработке PhotoYCC решалась задача максимального сохранения цветовых свойств, сравнимых с полученными при использовании высококачественной цветной фотоплёнки. Взаимное преобразование в цветовых пространствах R',G',B' в $PhotoY'C'C'$ данных, прошедших гамма-коррекцию, при нормировании R',G',B' в диапазоне 0–1 производят с помощью выражений [155]:

$$\begin{cases} Y' = 0.213R' + 0.419G' + 0.081B', \\ C1' = -0.131R' - 0.256G' + 0.387B' + 0.612, \\ C2' = 0.373R' - 0.312G' - 0.061B' + 0.537. \end{cases} \quad (1.13)$$

В связи с тем что возможности PhotoYCC-модели несколько шире, чем у RGB-модели, преобразование из первого цветового пространства во второе обычно производится с учётом специфики средств воспроизведения, например, компания Intel [155] предлагает следующий вариант с гамма-коррекцией в качестве возможного для электронно-лучевых мониторов:

$$\begin{cases} R' = 0.981Y + 1.315(C2 - 0.537), \\ G' = 0.981Y - 0.311(C1 - 0.612) - 0.669(C2 - 0.537), \\ B' = 0.981Y + 1.601(C1 - 0.612), \end{cases} \quad (1.14)$$

где уровни значений $Y,C1$ и $C2$ ограничены диапазоном 0–1.

YCoCg-модель была разработана для повышения эффективности сжатия цветных изображений, она содержит яркостную (Y) и две цветоразностные составляющие: Co – смещённый оранжевый и Cg – смещённый зелёный. Взаимное преобразование с RGB производится в соответствии с выражениями [158]:

$$\begin{cases} Y = \frac{1}{4}R + \frac{1}{2}G + \frac{1}{4}B, \\ Co = \frac{1}{2}R - \frac{1}{2}B, \\ Cg = -\frac{1}{4}R + \frac{1}{2}G - \frac{1}{4}B \end{cases} \quad (1.15)$$

и

$$\begin{cases} R = Y + Co - Cg, \\ G = Y + Cg, \\ B = Y - Co - Cg. \end{cases} \quad (1.16)$$

YCoCg-модель имеет несколько больший потенциал для отображения цветовых оттенков, чем RGB, поэтому при взаимных преобразованиях YCoCg в RGB часть информации может быть безвозвратно утрачена.

Модели цветового пространства HSV (Hue – оттенок, Lightness – яркость, Saturation – насыщенность) и HLS (Hue – оттенок, Saturation – насыщенность, Value – значение) разработаны как «интуитивно понятные» в манипуляциях с цветами и основаны на некоторых принципах восприятия цвета человеком. Для геометрической интерпретации данных цветовых пространств наиболее часто используют цилиндрические координаты.

Модель CIE *XYZ* разработана Международной комиссией по освещению. Компонент *Y* определяет яркость, а *X* и *Z* – оттенок цвета. Для геометрической интерпретации этой цветовой модели обычно используют прямоугольную систему координат. В положительную область пространства CIE *XYZ*, ограниченную значениями 0–1 по осям, могут быть отображены все видимые цвета. Кроме того, значительная часть цветов из куба, вписанного в пространство CIE *XYZ*, выходит за диапазон воспринимаемых человеком цветов.

Для отображения R', G', B' в $CIE\,XYZ$ (при ограничении значений X, Y, Z диапазоном 0–1) и обратно используют следующие выражения [166]:

$$\begin{cases} X = 0.412453R' + 0.35758G' + 0.180423B', \\ Y = 0.212671R' + 0.71516G' + 0.072169B', \\ Z = 0.019334R' + 0.119193G' + 0.950227B', \end{cases} \quad (1.17)$$

и

$$\begin{cases} R' = 3.240479X - 1.53715Y - 0.498535Z, \\ G' = -0.969256X + 1.875991Y + 0.041556Z, \\ B' = 0.055648X - 0.204043Y + 1.057311Z. \end{cases} \quad (1.18)$$

Рассмотренные взаимные преобразования цветовых пространств возможно интерпретировать как тензорные операции. Для преобразования пространств достаточно использовать операцию сложения и свёртки тензоров. Запишем операцию свёртки двух тензоров: $t_{im}v^m = d_i$, где t_{im} – тензор второго ранга, матрица преобразований; и тензоры первого ранга, векторы v^m и d_i – цветовых пространств. Следует отметить, что в приведённых преобразованиях цветовых пространств векторы v^m и d_i не являются взаимно контравариантным и ковариантным, т.е. $v^m \neq d^m$ и $d_i \neq v_i$, поскольку тензоры второго ранга t_{im} не являются метрическими. Напомним, что для метрического тензора должно выполняться условие $t_{im} = t_{mi}$, симметрия относительно главной диагонали матрицы.

1.5. Форматы цифровых изображений

Принципы построения современных вычислительных машин определяют физические основы преобразования информации в них. Световой импульс, передающий информацию по оптоволоконному кабелю; заряд, хранящийся в ячейке памяти; или токи, протекающие через логические элементы интегральных схем, – всё это, с точки зрения современной физики, разновидности электромагнитного поля. Различные виды преобразования электромагнитной энергии являются основой современной цифровой техники. Большинство вычислительных машин работает с дискретными величинами, представленными при помощи особым способом модулированного электромагнитного поля. Можно заметить, что наряду с цифровыми вычислительными машинами существуют *аналоговые* (АВМ).

Основной единицей информации в цифровых ЭВМ является *бит*, на базе которого формируются более крупные единицы информации. Бит – это неделимый квант информации, способный принимать лишь одно из двух состояний: «1» или «0».

Используемые для ЦОС вычислительные системы строятся на базе цифровых сигнальных процессоров, которые, в свою очередь, можно разделить на процессоры с фиксированной и плавающей точкой. Различие между этими разновидностями вычислительных устройств состоит в возможности обрабатывать различные числовые типы на аппаратном уровне. Первые предназначены для обработки целочисленных форматов, при этом они могут обрабатывать форматы чисел с плавающей точкой за счёт специальных подпрограмм. Процессоры с плавающей точкой могут обрабатывать данные в формате с плавающей точкой и целочисленные (например, для организации счётчиков). Многие современные процессоры с фиксированной точкой имеют гибридную структуру и содержат интегрированные (математические)

сопроцессоры, позволяющие производить вычисления в формате с плавающей точкой.

Отдельного внимания заслуживают технологии параллельных вычислений. Обобщённая модель классификации по признакам параллелизма в вычислительной архитектуре предложена М. Флинном в 1970-х гг. (табл. 1.1).

Таблица 1.1. Классификация параллелизма в вычислительных системах по М. Флину

Поток данных	Поток команд	
	одиночный (Single Instruction)	множественный (Multiple Instruction)
Одиночный (Single Data)	ОКОД (SISD)	МКОД (MISD)
Множественный (Multiple Data)	ОКМД (SIMI)	МКМД (MIMD)

Многие современные вычислительные системы в той или иной степени используют технологии параллелизма, поэтому однозначно отнести конкретную ЭВС к какому-либо типу (см. табл. 1.1) не всегда возможно [16, 127].

В большинстве современных компьютеров основной массив оперативной памяти представляет собой линейное адресное пространство и имеет одно адресное измерение, поэтому одним из базовых типов данных, непосредственно связанных с архитектурой вычислительных машин, является массив. Адресное пространство линейно, при этом большинство цифровых устройств может одновременно обращаться к различным адресам памяти. Массив позволяет упорядоченно располагать в памяти вычислительной машины данные, состоящие из множества элементов одного типа, индексируя каждый элемент идентификатором. В самом простом случае элементом массива является ячейка памяти с уникальным адресом, адрес ячейки соотносится с индексом элемента массива с учётом смещения (табл. 1.2).

Таблица 1.2. Расположение данных в памяти ЭВМ

Индекс массива a[]	Адрес	Значение
…	…	…
a[2]	0x00E39732	0x01
a[1]	0x00E39731	0xFF
a[0]	0x00E39730	0xA0
…	…	…

В случае одномерных цифровых сигналов номер отсчёта сигнала соответствует адресу ячейки памяти, с учётом смещения и разрядности (формата) хранения сигнала в памяти ЭВМ. Если сигнал многомерный, то его необходимо отображать на одномерную память, при такой процедуре применяют многомерные массивы, или массивы массивов.

Как было показано ранее, цифровые сигналы удобно рассматривать в виде тензорного поля, причём в большинстве случаев тензор имеет ранг «нуль» (скаляр) или «единица» (вектор). Если тензор имеет нулевой ранг, значение элемента сигнала сохраняется в виде единственного числа, если единичный – в виде массива или структуры.

Рассмотрим пример создания структуры описания цветового пространства (листинг 1.1), которую можно рассматривать как тензор первого ранга – в трёхмерных цветовых пространствах, например RGB и YCbCr. Директива препроцессора `#pragma pack (push, 1)` устанавливает плотность упаковки данных в структуре в 1 байт, а директива `#pragma pack (pop)` возвращает установки компилятора по умолчанию. В связи с тем что в языке C++ для трёхбайтных элементов не предусмотрен специальный тип, в случае использования объединений `(union)` в памяти будет помещено объединение длиной в четыре, при типе `long int` – в шесть байт[4] [100]. Таким образом, по адресу элемента объединения хранятся четырёхбайтное целое представление цветового значения элемента (тензор нулевого ранга) и покомпонентная структура `ColorSpace` (тензор первого ранга), такой подход позволяет проводить быстрые преобразования типов данных, а следовательно, и тензорных полей цифровых сигналов. Для хранения полноцветного изображения требуется 24 бита на каждый пиксел, в примере (листинг 1.1) задействовано 32 бита, для экономии адресного пространства (используется только 24 битов на пиксел) достаточно исключить `union`.

Листинг 1.1. Структура вектора цветового пространства RGB и YCbCr (C++)

```
//=================================================================
typedef unsigned char uchar;       // беззнаковый байт
typedef unsigned long int  uint;   // беззнаковый целый (int)
#pragma pack (push, 1)
// цветовая палитра RGB
// покомпонентное представление, в скобках указаны компоненты
// палитр (RGB) и (YCbCr) соответственно
struct ColorSpace {
  uchar byteLo;  //  компонент  (R)  (Y)является  младшим  байтом
(Lower)
  uchar byteAv;  //  компонент  (G)  (Cb)является  средним  байтом
(Average)
```

[4] Отметим, что для разных компиляторов и вычислительных платформ длина типов может различаться, поэтому для определённости часто используют явное указание длины, например, вместо `int` применяют тип `int32`.

```
  uchar byteHi; // компонент (B) (Cr) является старшим байтом
(Hight)
};
// объединение для быстрого преобразования вектора в скаляр и об-
ратно
union UColorSpace {
  ColorSpace rgb;
  uint numb;
};
#pragma pack (pop)
//=============================================================
```

Рассмотрим листинг программы выделения памяти для поля цифрового сигнала (листинг 1.2). В качестве примера возьмём статическое изображение с количеством оттенков 16 777 215 (0xFFFFFF), для сохранения элементов изображения будем использовать ранее созданную структуру `ColorSpace` (см. листинг 1.1).

Листинг 1.2. Выделение памяти для полноцветного изображения (C++)

```
//=============================================================
// ...
// выделяем память
#pragma pack (push, 1)
// объявляем указатель на двумерный массив изображения
ColorSpace** data=0;
// iHeight - высота изображения (число строк)
// iWidth  - ширина изображения (длина строк (столбцы))
data=new ColorSpace*[iHeight];
for (int i=0; i<iHeight; i++) {
  data[i]=new ColorSpace[iWidth];
  memset(data[i], 0, iWidth* sizeof(ColorSpace)); // инициализация
}
#pragma pack (pop)
// ...
// используем выделенную память
// ...

// чистим память
for (int i=0; i<iHeight; i++) {
  delete[]data[i];
}
delete[]data;
data=0;
// ...
//=============================================================
```

Дефрагментация, возникающая при многократном выделении и очистке памяти, может способствовать утечке ресурсов и снижению быстродействия. Поэтому следует использовать существующие в рамках стандартной библиотеки (C++) интеллектуальные указатели и алгоритмы

сборки мусора, позволяющие эффективно использовать адресное пространство ЭВМ.

1.6. Программные преобразования цветовых пространств

Взаимные преобразования цветовых пространств выполняют с помощью как программных, так и аппаратных средств. Рассмотрим вариант программного преобразования на примере RGB в YCbCr (вариация $Y_{709}CbCr$) в соответствии с выражением (1.11), на базе операций с плавающей точкой (листинг 1.3), используемый тип данных `ColorSpace` подробно рассмотрен ранее (см. листинг 1.1).

Листинг 1.3. Преобразование значения из цветового пространства RGB в YCbCr, вычисления в формате с плавающей точкой (C++)

```
//=================================================================
ColorSpace x;  // объявляем структуру типа ColorSpace
...
// с плавающей точкой
float R, G, B;
R=(float)x.byteLo;
G=(float)x.byteAv;
B=(float)x.byteHi;
// компонент Y
x.byteLo=(uchar)floor(0.183*R+0.614*G+0.062*B+16);
// компонент Cb
x.byteAv=(uchar)floor(-0.101*R-0.338*G+0.439*B+128);
// компонент Cr
x.byteHi=(uchar)floor(0.439*R-0.399*G-0.040*B+128);
...
//=================================================================
```

Очевидно, что преобразование, выполненное подобным образом, будет не самым эффективным решением с точки зрения использования вычислительных ресурсов. Требуются двойное преобразование форматов из `unsigned char` в `float` и обратно, три умножения и три сложения, операция округления в формате с плавающей точкой для каждого из каналов. Как будет показано далее, подобные преобразования возможно значительно оптимизировать за счёт применения вычислений на базе формата с фиксированной точкой.

1.7. Модель классификации изображений

Разнообразие цифровых графических изображений, форматов их хранения, способов визуализации и обработки требует упорядочивания по общим и отличительным признакам. Классификация графических объектов является непременным фактором выбора оптимальных средств и методов их обработки, хранения и анализа.

В первую очередь, все цифровые изображения можно разделить на две группы: статические, т.е. неизменные во времени, и динамические – меняющиеся. Динамические изображения обычно называют видеорядом или компьютерной анимацией (частный случай компьютерной графики). Большинство динамических изображений формируется из совокупности упорядоченных кадров, при этом каждый кадр является статическим изображением.

Статические и динамические изображения могут иметь различную размерность. В действительности прообразы большинства сигналов являются бесконечномерными явлениями. До сих пор не найден эффективный способ описания структуры таких комплексных явлений [117], поэтому на практике применяют значительно упрощённые, ограниченные техническими возможностями и необходимостью, способы отображения реальных прообразов в образы цифровых сигналов. В зависимости от размерности сигнала различают 1D, 2D, 3D и т.д. изображения. В динамических изображениях одно или несколько измерений соответствует времени. Современные графические системы также работают с изображениями 4D, 5D и более [123]. Причиной увеличения мерности изображения является стремление к реалистичности передачи визуальной информации. Например, к классу 4D относят динамические трёхмерные изображения. Дальнейшее увеличение мерности определяется числом и степенью свободы оптических регистраторов изображения. Различают способы формирования изображения с одной (single-view) и с множества точек (multi-view). Достаточно продолжительное время разрабатываются системы визуализации трехмерных статических и динамических изображений в пространстве. Существует ряд решений, которые возможно применять для практических целей, однако сдерживающим фактором является колоссальный объём информации, необходимой для генерации многомерных изображений – образов реального мира. В современных условиях широко распространены псевдотрёхмерные изображения и устройства визуализации простых графических объектов в трёхмерном виде (3D и 4D). Псевдотрёхмерные изображения формируются за счёт особенностей стереоскопического зрения человека. Основой обычно служит двумерное изображение, которое расщепляется на две части при помощи специальных оптических устройств или методов цифровой обработки сигналов.

По особенностям цветопередачи изображения можно разделить на:

– монохроматические, каждый пиксел такого изображения представляется одним битом;

– полутоновые, каждый пиксел изображения обычно может иметь 2^n значений от 0 до $2^n - 1$, обозначающих оттенки серого или иного цвета, в ряде систем некоторые значения из этого диапазона могут нести специаль-

ную информацию, например, о мигании пиксела с заданной частотой или прозрачности (т.н. альфа-канал);

– цветные изображения с различными цветовыми схемами позволяют передавать информацию о цвете: различают дискретно-тоновые изображения (число цветовых оттенков существенно ограничено) и с непрерывным тоном (значительное число оттенков – распространённые цифровые цветовые схемы позволяют передать до 16 777 216 цветовых оттенков, что при просмотре создаёт впечатление непрерывности цветовой палитры).

По способу представления графических объектов изображения можно разделить на:

– векторные, когда изображение рассматривается как совокупность графических примитивов (такие изображения характерны для научно-технической документации, компьютерной графики);

– растровые, когда изображение рассматривается как совокупность отдельных пикселов, расположенных в определённом порядке (такие изображения характерны для цифровых фотографий).

Возможна классификация изображений по наличию и уровню шума, например, для цифровых фотографий характерен шум, а синтетические изображения, полученные при помощи компьютерной графики, могут вовсе не содержать шумов. Характер шума также может различаться, например, шумы на снимках ЯМР-томографа и фотографии космической обсерватории. Шумы могут возникать на различных этапах преобразования информации, поэтому способ и условия получения изображения определяют характер шумов.

Графические изображения можно классифицировать по способу цветопередачи. Многие снимки изначально выполнены в невидимом спектре, например инфракрасном, ультрафиолетовом, рентгеновском, радиочастотном. Для визуализации таких снимков применяют функциональное преобразование образов, полученных в невидимом диапазоне, к образам в видимом диапазоне. Цвета в таких изображениях называются ложными и соответственно говорят «изображения в *ложных цветах*» (*false color*), про изображения в реальных цветах говорят «в *истинных цветах*» (*true color*). В некоторых случаях замену цветов выполняют из-за особенностей технологического процесса или для лучшего восприятия информации.

Существуют классификации по способу получения и области применения изображений, смысловой нагрузке: аэрофотоснимки, космические фотографии, медицинские, чертежные, текстовые, кинематографические, мультипликационные и пр.

В соответствии с типом изображения и его особенностями применяют различные методы обработки и сжатия, используют разные форматы хранения, принципы визуализации. Методы работы с изображением и ряд

других количественных и качественных характеристик можно рассматривать как классификационные признаки.

Наиболее сложные системы классификации изображений, или *системы компьютерного зрения*, являются частью области знаний, относящейся к *распознаванию образов*.

2. Учёт ресурсов, разработка и оптимизация программ

Несмотря на гиперскоростное развитие в сфере информационных технологий за последние 50–60 лет, актуальным остается высказанное в 1995 г. замечание Н. Вирта: «программы становятся медленнее более стремительно, чем компьютеры становятся быстрее». С учётом постоянно растущего объёма данных, требующих обработки, и более жёстких требований к функциональным возможностям программного обеспечения оптимизация программного кода является очень актуальным направлением современных исследований. В основе работы любой вычислительной машины лежит заданная последовательность действий, называемая алгоритмом.

Алгоритм – это точный набор инструкций, определяющих способ решения некоторой задачи. Алгоритм обладает следующими свойствами:

– дискретность – содержит конечное число определённых действий (шагов);

– детерминированность – в каждый момент времени его состояние (текущее действие) должно однозначно определяться;

– исполнимость – алгоритм должен содержать только те инструкции, которые может реализовать исполнитель;

– результативность – работа алгоритма должна заканчиваться получением определённых результатов, являющихся решением задачи с корректно поставленными условиями (исходными данными);

– массовость – алгоритм должен обеспечивать решение некоторого класса сходных задач, т.е. должен быть применим к различным наборам исходных данных, удовлетворяющих условиям задачи, на решение которой ориентирован.

Способ – определённый порядок последовательных действий, направленных на достижение цели и обладающих всеми свойствами алгоритма: дискретность, определённость (детерминированность), результативность, массовость. Обычно алгоритм является более формальным, с точки зрения обозначений, выражением способа.

Метод – совокупность способов, объединённых некоторой методологией (концептуальным обобщением) и направленных на достижение цели. Соответственно метод также обладает всеми свойствами алгоритма.

2.1. Вычислительная сложность алгоритмов

Машинно-независимые алгоритмы разрабатываются на гипотетическом компьютере, так называемой *машине с произвольным доступом к памяти* (*Random Access Machine*), или *RAM-машине* [110]. Принципы и условия работы RAM-машины следующие:

– любая простая команда исполняется в точности за один шаг;

– циклы, подпрограммы не считаются простыми командами и требуют некоторого конечного числа шагов (состоят из конечного числа простых операций);

– каждое обращение к памяти занимает ровно один шаг, объем памяти RAM-машины не ограничен, всё адресное пространство линейно и полностью доступно в любой момент времени исполнения программы.

С помощью RAM-машины можно оценить *вычислительную сложность* алгоритма как число шагов, требуемое для решения задачи при любом наборе данных. Время исполнения алгоритма RAM-машиной вычисляется по общему количеству шагов исполнения программы. RAM-машина является моделью (*RAM-моделью*) реальных вычислительных машин, поэтому позволяет произвести только приблизительную оценку вычислительной сложности алгоритма. В отличие от абстрактных RAM-машин практически используемые вычислительные машины являются сложными техническими устройствами с большим разнообразием архитектур, работающими в многозадачных режимах и производящими параллельные вычисления. Поэтому для наиболее эффективной реализации алгоритма конкретной ЭВМ требуется доработка, а иногда – полная переработка с целью оптимизации вычислений и использования ресурсов. Следовательно, процедуры оценки вычислительной сложности алгоритма для RAM-машин и реальных ЭВС различаются.

Принято рассматривать три оценки *вычислительной сложности* алгоритма на входном наборе данных размером n:

– *наихудший случай* – функция, определяемая максимальным количеством шагов, требуемых для завершения алгоритма;

– *наилучший случай* – функция, определяемая минимальным количеством шагов, требуемых для завершения алгоритма;

– *средний случай* – функция, определяемая средним количеством шагов, требуемых для завершения алгоритма.

Как для различных входных наборов данных размером $n = \text{const}$ может быть различная вычислительная сложность? Рассмотрим пример решения системы уравнений:

$$\begin{cases} a_0 x_0^2 + b_0 x_0 + c_0 = 0, \\ a_1 x_1^2 + b_1 x_1 + c_1 = 0, \\ \ldots \\ a_{n-1} x_{n-1}^2 + b_{n-1} x_{n-1} + c_{n-1} = 0, \end{cases}$$

где $D_i = b_i^2 - 4a_ic_i$, $i = 0,...,n-1$, при $D_i \geq 0$ требуются действия с действительными числами, а при $D_i < 0$ – с комплексными. Следовательно, на входном наборе данных размером $n = \text{const}$ понадобится различное число действий, например, при $D_i \geq 0, i = 0,...,n-1$ вычислительная сложность будет меньше, чем при $D_i < 0, i = 0,...,n-1$. Отметим, что мы лишь сравниваем вычислительную сложность, не используя критерии «наилучший» или «наихудший», т.к. наилучший случай может быть реализован в конкретном алгоритме только для некоторого множества наборов данных: например, в для $a_i = 1, b_i = 0, c_i = 0, i = 0,...,n-1$ может быть реализован наилучший вычислительный случай $x_i = 0,\ i = 0,...,n-1$.

Вычислительную сложность можно выразить с помощью функций, зависящих от размеров входного набора данных n. Для различных алгоритмов сложность таких функций может значительно различаться, кроме того, функции могут содержать множественные условия, зависящие от набора данных, в общем случае применение таких функций на практике затруднено. Вычислительная сложность $f(n)$ оценивается при помощи асимптотических функций, записываемых в асимптотической нотации ($O(\ldots)$, $\Omega(\ldots)$, $\Theta(\ldots)$), которые формально определяются [110] следующим образом:

– $f(n) = O(g(n))$ означает, что существует функция $f(n)$, *ограниченная сверху* функцией $g(n)$, с точностью до постоянного множителя $c = \text{const}$, иными словами, существует такое значение c, для которого $f(n) \leq c \cdot g(n)$ при достаточно большом значении n;

– $f(n) = \Omega(g(n))$ означает, что функция $f(n)$ *ограничена снизу* функцией $g(n)$, с точностью до постоянного множителя $c = \text{const}$, иными словами, существует такое значение c, для которого $f(n) \geq cg(n)$ при достаточно большом значении n;

– $f(n) = \Theta(g(n))$ означает, что функция $f(n)$ *ограничена сверху* функцией $g(n)$ и *ограничена снизу* функцией $g(n)$, с точностью до постоянных множителей $c_1 = \text{const}$ и $c_2 = \text{const}$ соответственно. Иными словами, существуют такие значения c, для которых $c_2 g(n) \leq f(n) \leq c_1 g(n)$ при достаточно большом значении n.

В основном наибольший интерес представляет $f(n) = O(g(n))$ – верхняя граница (рис. 2.1), на основе данной функции выполняют сравнение алгоритмов, оценку требуемых вычислительных мощностей и т.п.

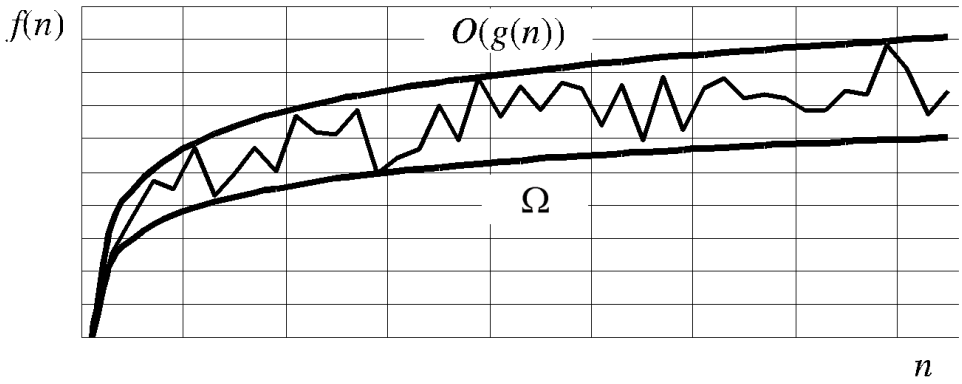


Рис. 2.1. Верхняя и нижняя границы
функции вычислительной сложности алгоритма $f(n)$

В табл. 2.1 приведены данные о времени вычислений на ЭВС, выполняющей 1 миллиард операций в секунду, т.е. время исполнения одной операции 10^{-9} с (отметим, что один год соответствует $\approx 3 \cdot 10^7$ с).

Таблица 2.1. Оценка времени вычислений для некоторых распространённых асимптотических функций

n	$\log_2(n)$, с	$n \cdot \log_2(n)$, с	n^2, с	n^3, с	2^n, с
10^1	$4 \cdot 10^{-9}$	$33 \cdot 10^{-9}$	$100 \cdot 10^{-9}$	10^{-6}	10^{-6}
10^2	$7 \cdot 10^{-9}$	$664 \cdot 10^{-9}$	$10 \cdot 10^{-6}$	10^{-3}	10^{21}
10^3	$10 \cdot 10^{-9}$	$9.97 \cdot 10^{-6}$	10^{-3}	1	10^{292}
10^4	$14 \cdot 10^{-9}$	$132.9 \cdot 10^{-6}$	0.1	10^3	10^{3001}
10^5	$17 \cdot 10^{-9}$	$1.66 \cdot 10^{-3}$	10	10^6	10^{30093}
10^6	$20 \cdot 10^{-9}$	$19.93 \cdot 10^{-3}$	10^3	10^9	10^{301020}

Сократить время исполнения вычислительных задач возможно путем оптимизации вычислительных алгоритмов или повышения быстродействия ЭВС.

Объединение высокоскоростной магистралью множества высокопроизводительных процессоров позволяет получать системы, называемые *суперкомпьютерами*. Быстродействие в таких системах достигается за счёт распараллеливания задачи на множество подзадач, соответственно требуются специальные алгоритмы и методы, на базе которых создаётся программное обеспечение для суперкомпьютеров. Для оценки реального быстродействия ЭВС применяют внесистемную единицу Flops (Floating point OPerations per Second), показывающую, сколько операций с плавающей запятой в секунду выполняет данная вычислительная система.

Следует отметить, что в своих работах Бремерманн [137] показал существование некоторого вычислительного предела и трансвычислительных задач, для решения которых требуется обработка более чем 10^{93} бит.

Решение трансвычислительных задач может быть осуществлено компьютером размером с Землю, время вычисления будет равно времени ее существования. Существование практических трансвычислительных задач показывает, что имеется некоторое множество проблем, которые невозможно преодолеть только за счёт наращивания вычислительной мощности. Возможно, решение таких задач станет реальным с развитием математики, появлением новых вычислительных методов или совершенно новых областей знаний, использующих неизвестные пока принципы получения и обработки информации. Возможно, решение данных задач лежит в области иных концептуальных подходов, доступных в будущем, в соответствии с которыми трансвычислительные задачи в современной постановке могут, например, вовсе утратить свою актуальность.

2.2. Измерение машинного времени

Практический интерес при реализации алгоритмов цифровой обработки сигналов и вычислениях представляет время расчётов на вычислительной машине. Значительно варьирующаяся архитектура вычислительных машин не позволяет получить универсальной зависимости машинного времени от вычислительной сложности, ведь современные машины позволяют распараллеливать решаемые задачи, производить вычисления, применяя специальные сопроцессоры. В свою очередь, учет особенностей архитектуры компьютера, использование специальных процессорных инструкций позволяет добиться наилучших результатов оптимизации алгоритмов и программ для вычислительного устройства определённого типа. Особенно критично время вычислений при обработке сигналов в реальном масштабе времени, например, в операционных системах реального времени или специальных аппаратно-программных комплексах [34, 80, 92].

Таким образом, время вычисления одной и той же задачи сильно зависит от многих факторов. Поэтому для оценки практических затрат машинного времени применяют измерения. В частности, разработаны специальные тесты, позволяющие измерять время вычислений. Следует отметить, что многие тесты разработаны самими производителями вычислительных устройств (процессоров, чипсетов, сопроцессоров, памяти и др.), одной из целей таких тестов является демонстрация преимущества своих изделий над аналогами. Этот фактор может повлиять на объективность выводов, поэтому для более точной оценки желательно использовать вычислительные тесты от различных производителей, учитывающие архитектуру и особенности вычислительного устройства.

Машинное время обычно определяется как разность времени начала и времени завершения расчётов. Однако такой подход не всегда может

дать приемлемый результат, особенно для современных сложных вычислительных систем. Дело в том, что большинство вычислительных систем являются многозадачными и могут выполнять некоторое множество расчётов одновременно или создавать видимость такой одновременности, квантуя машинное время и распределяя его между конкурирующими процессами. Ситуации, когда все вычислительные ресурсы предоставлены одному процессу и необходимо определить время его выполнения, достаточно тривиальны. Поэтому рассмотрим некоторые особенности расчёта машинного времени в многозадачных операционных системах (ОС) на примере ОС семейства UNIX и Windows. Внутри вычислительной системы «существует» несколько «времён»: *пользовательское* (затраченное на исполнение машинных инструкций, образующих процесс), *системное* (затраченное на выполнение машинных инструкций ядром от имени процесса, т.е. на исполнение системных вызовов процесса) и машинное (время работы вычислительной системы). Сумму пользовательского и системного времени часто называют *процессорным временем* [115].

В ОС UNIX любой процесс может вызвать функцию `times(struct tms *buffer)`, параметром которой является указатель на структуру `tms` (листинг 2.1). Функция `times` возвращает *общее время*, соответствующее времени от момента запуска процесса.

Значения типа `clock_t` содержат время, измеряемое в тактах (ticks). Для перевода этого значения в секунды необходимо воспользоваться функцией `sysconf`, с помощью вызова с параметром `_SC_CLK_TCK` можно получить количество тактов в секунду (частоту) для данной платформы.

Листинг 2.1. Структура `tms` (UNIX, C)

```
//=============================================================
struct tms {
  clock_t tms_utime;  // пользовательское время
  clock_t tms_stime;  // системное время
  clock_t tms_cutime; // пользовательское время для завершения
                      // потомка
  clock_t tms_cstime; // системное время для завершившегося
                      // потомка
}
//=============================================================
```

В ОС Windows информацию о временных характеристиках процесса можно получить с помощью функции `GetProcessTimes`, о временных характеристиках потока – с помощью `GetThreadTimes` (в Windows 9x отсутствуют), о времени системы – с помощью `GetSystemTime` (листинг 2.2). Функции `GetProcessTimes` и `GetThreadTimes` в случае успеха возвращают значение, отличное от нуля, при неудаче код ошибки можно узнать при

помощи функции `GetLastError`, функция `GetSystemTime` ничего не возвращает [120, 122, 161].

Листинг 2.2. Функция `GetProcessTimes` (Windows, C++)

```
//=================================================================
BOOL WINAPI GetProcessTimes(
  HANDLE hProcess,            // дескриптор процесса
  LPFILETIME lpCreationTime, // указатель на структуру FILETIME,
                              // которая
                              // получает время создания процесса.
  LPFILETIME lpExitTime,      // Указатель на структуру FILETIME,
                              // которая
                              // получает время выхода из потока.
                              // Если выход из потока не выполнен, то
                              // содержание
                              // этой структуры не определено.
  LPFILETIME lpKernelTime,    // Указатель на структуру FILETIME,
                              // которая
                              // получает значение системного
                              // времени, т.е.
                              // выполнения процесса в режиме ядра.
  LPFILETIME lpUserTime       // Указатель на структуру FILETIME,
                              // которая
                              // получает значение пользовательского
                              // времени, т.е. времени выполнения
                              // процесса в
                              // пользовательском режиме.
);
// Функция GetThreadTimes
BOOL WINAPI GetThreadTimes(
  HANDLE hThread,             // дескриптор потока
  LPFILETIME lpCreationTime, // указатель на структуру FILETIME,
                                 которая
                              // получает время создания потока.
  LPFILETIME lpExitTime,      // Указатель на структуру FILETIME,
                              // которая
                              // получает время выхода из потока.
                              // Если выход из потока не выполнен, то
                              // содержание этой структуры не
                              // определено.
  LPFILETIME lpKernelTime,    // Указатель на структуру FILETIME,
                              // которая
                              // получает количество системного
                              // времени, т.е.
                              // выполнения потока в режиме ядра.
  LPFILETIME lpUserTime       // Указатель на структуру FILETIME,
                              // которая
                              // получает количество пользовательского
                              // времени, т.е. выполнения потока в
                              // пользовательском режиме.
);
// Функция GetSystemTime
```

```
void GetSystemTime(
  LPSYSTEMTIME lpSystemTime  // Указатель на структуру SYSTEMTIME,
                             // которая
                             // получает текущую дату и время
                             // системы.
);
// 64-битная структура FILETIME и указатель *PFILETIME
typedef struct _FILETIME {
  DWORD dwLowDateTime;       // 32 младших бита
  DWORD dwHighDateTime;      // 32 старших бита
} FILETIME,  *PFILETIME;
//=============================================================
```

Используя описанные системные функции, можно получить объективные характеристики времени выполнения вычислительного процесса или его части. На основе полученных данных возможно делать выводы о затратах машинного времени для осуществления расчётов.

Следует отметить, что использование рассмотренных в листингах 2.1 и 2.2 функциональных средств не может заменить полноценного анализа программы, выполняемого разработчиком-оптимизатором с помощью профилировщиков – специальных приложений для анализа производительности программ; отладчиков – программ, позволяющих контролировать и управлять исполнением кода; дизассемблеров – программ, позволяющих преобразовать машинный код в набор инструкций языка ассемблера.

2.3. Цифровые сигнальные процессоры

Снижению времени вычислений и вычислительных затрат в общем способствует специализация различных вычислительных машин на аппаратном уровне. Для преобразования цифровых сигналов предназначены *цифровые сигнальные процессоры* (ЦСП, Digital Signal Processor). Отличительными особенностями ЦСП являются [74]:

– специальные процессорные команды, выполняющие за один такт операцию сложения с накоплением;

– специфическая архитектура, в особенности разделение памяти данных и программ в соответствии с гарвардской или супергарвардской архитектурой;

– возможность векторно-конвейерной обработки при помощи генераторов адресных последовательностей и кольцевых буферов в памяти, предоставляющих непрерывный доступ к обновляющимся данным и исключающих переполнение;

– определённое время (в тактах) выполнения команд и их последовательностей, что позволяет встраивать процессоры в системы реального времени и создавать кластеры ЦСП;

– организация циклов с автоматической проверкой условия завершения;

– возможность одновременной выборки двух операндов;

– таблицы готовых значений некоторых функций, например, sin(), cos() и т.д.

Следует отметить, что современные тенденции развития микропроцессорной техники направлены на интеграцию различных архитектур и концепций на базе одного кристалла. Например, современные процессоры общего назначения, выпускаемые корпорациями Intel и AMD, содержат интегрированные сопроцессоры (графические, математические), специальные наборы инструкций MMX, SSE (SSE2, SSE3) и 3DNow (3DNow2) соответственно, реализующие возможности ЦСП.

Наиболее широко ЦСП применяются для решения следующих задач:

– кодирование, мультимедийная обработка аудио- и видеоданных;

– производство и проектирование техники (анализаторы спектра, генераторы сигналов, системы локации, робототехника и др.);

– распознавание образов (речи, звуков, графических образов и др.);

– многоскоростная обработка цифровых сигналов.

Реализации ЦСП различаются типом обрабатываемых данных: с фиксированной или с плавающей точкой. На практике различие типов данных проявляется в значении отношения сигнал/шум. Для формата с плавающей точкой характерен приблизительно в 3000 раз меньший шум квантования, чем для формата с фиксированной точкой [112].

Разрядность ЦСП также определяет возможности процессора (наиболее распространены 16- и 32-разрядные ЦСП). Быстродействие ЦСП зависит от ряда факторов, связанных со временем обращения к памяти и временем выполнения машинных инструкций (арифметических и логических операций). Немаловажным фактором, определяющим конечное время выполнения тех или иных задач, является оптимизация программного кода под конкретный вид процессора. Производители ЦСП предоставляют библиотеки оптимизированного программного кода, ориентированные на выполнение множества стандартных задач (свёртка, быстрое преобразование Фурье и др.).

Возможно синтезировать многопроцессорные системы при помощи объединения нескольких ЦСП с выделенными линиями связи «точка-точка» через общую глобальную память или при помощи ячеистых структур.

2.4. Графические процессоры

Графические процессоры ГП (Graphics Processing Unit, GPU) – специфические вычислительные устройства, ориентированные на графический *рендеринг* (синтез изображений). Вычислительная мощность новейших ГП обеспечила новые возможности их применения – для неграфических вычислений, например, многопоточное решение систем дифференциальных уравнений.

Первые графические процессоры, работающие в составе персональных вычислительных машин, выполняли функцию буфера кадров, периодически выводя отображение определённой области памяти на экран монитора (само изображение формировалось центральным процессором). Поскольку во многих случаях центральный процессор не мог обработать всю необходимую графическую информацию за время между обновлениями кадров, предварительно кадр (или его часть) формировался в буфере, не пересекающемся с отображаемой областью видеопамяти, а затем готовые данные копировались в видеопамять. Такой подход применяется во многих системах компьютерной графики.

Совершенствование вычислительных технологий и спрос на высококачественные встроенные графические системы определили развитие ГП как самостоятельную технологию решения специфических задач синтеза изображений. Основной целью ГП является разгрузка центрального процессора и расширение графических возможностей персонального компьютера.

Архитектура современных ГП значительно варьирует. Основной парадигмой их построения является распараллеливание вычислений. Синтезирование графической информации требует обработки значительных объёмов данных, причём в силу специфики данные могут быть обработаны в параллельном режиме ограниченным числом команд. Таким образом, основу архитектуры ГП составляет множество работающих параллельно простых вычислительных ядер. Совместимость различных ГП достигается с помощью специального интерфейсного программного обеспечения (драйверов графических устройств). Дополнительные программные библиотеки (OpenGL, DirectX) [20] позволяют связывать низкоуровневый программный и аппаратный интерфейс графического устройства с системой функций синтеза и обработки двух- и трёхмерных графических примитивов.

Значительная мощность ГП позволила использовать их для неграфических вычислений (General-purpose Graphics Processing Units, GPGPU), активно развивающихся благодаря повышению арифметической точности вычислений на ГП. Технологии неграфических вычислений представлены достаточно большим набором программных интерфейсов:

– OpenCL – фреймворк от разработчиков Apple Inc., Khronos Group (http://www.khronos.org/opencl/);

– CUDA (Compute Unified Device Architecture) – решение от Nvidia (http://www.nvidia.com/);

– DirectCompute – составная часть DirectX от Microsoft (http://www.microsoft.com/);

– AMD FireStream – технология неграфических вычислений от компании AMD (http://www.amd.com/).

Поддержка неграфических вычислений на базе ГП лидерами индустрии информационных технологий во многом определяет перспективы данного направления. Следует отметить, что пакет MATLAB (с 2010 г.) поддерживает технологию неграфических вычислений на ГП в компоненте Parallel Computing Toolbox [159].

2.5. Модели разработки программного обеспечения

Процессор, т.е. вычислительное ядро ЭВС, управляется потоком машинных кодов, образованных последовательностью нулей и единиц. Первичной моделью, или «семантическим интерфейсом» разработки, программного обеспечения является язык ассемблера. Ассемблер обеспечивает взаимно-однозначное соответствие между операторами и командами процессора. Ассемблер, при высокой квалификации разработчика, позволяет получить наиболее эффективный код для данного процессора, при переносе которого на другую вычислительную платформу (например, построенную на базе процессора того же типа, но с другой моделью адресации интерфейсных портов или ОЗУ) может потребоваться существенная доработка кода, вплоть до написания всей программы заново. С учётом многообразия процессоров, вычислительных архитектур и операционных систем, постоянной конкуренции разработчиков программного обеспечения применение ассемблера является очень затратным технологическим процессом. Однако следует отметить, что во многих разработках в сфере оборонной промышленности, бортовых вычислительных устройств авионики и космонавтики, в микрокомпьютерах промышленных и бытовых устройств успешно используется ассемблер. Таким образом, в индустрии программного обеспечения применение ассемблера могут себе позволить только очень консервативные разработчики, удаётся удерживающие рынок сбыта за счёт добротных, компактных и быстрых программ или владения монопольным правом на их производство. Некоторые задачи возможно решить только средствами ассемблера. Например, большинство компиляторов языков C++, Fortran не могут «напрямую» использовать технологии SIMD (*Single Instruction, Multiple Data* – одна команда,

множество источников данных), исключение составляет компиляторы от Intel [28].

Для снижения трудозатрат на разработку, отладку и оптимизацию вместо ассемблера используют языки более высокого уровня с целью создания более «человекопонятного» программного кода с меньшей зависимостью от архитектуры конкретной ЭВС. На сегодняшний день существует множество альтернативных подходов – от применения низкоуровневых языков до транслируемых языков разработки и виртуальных машин, практически абстрагированных от конкретной платформы.

Рассмотрим несколько различных уровней управления ЭВС:

– непосредственное управление ЭВС набором машинных кодов;

– применение ассемблера с последующей компиляцией в машинный код;

– применение низкоуровневых языков (C, C++, Fortran, Pascal и др.) с последующим преобразованием для конкретной операционной системы и аппаратной платформы;

– применение высокоуровневых транслируемых языков (BASIC, PHP и др.), которые переводятся в машинный код непосредственно в момент исполнения;

– применение специальных виртуальных вычислительных сред (Matlab, Matematica, Scilab, FreeFem++, Ch SoftIntegration и др.).

Применение виртуальных машин позволяет создавать требуемую виртуальную аппаратную и операционную среду, в рамках которой возможно использовать любой из перечисленных подходов.

Рассмотрим процесс разработки программного обеспечения как технологический цикл. Принято выделять несколько моделей жизненного цикла программного обеспечения. Разработка программного обеспечения может иметь *каскадный* или *итерационный* (циклический) характер. При каскадном сначала формируются требования (техническое задание на разработку) в полном объёме, затем осуществляется поэтапная разработка. Причём завершение этапа означает переход к последующему этапу без возврата к предыдущему, что, вообще говоря, во многих случаях достаточно затруднительно. Особенные сложности такой подход вызывает в крупномасштабных проектах.

В соответствии с ГОСТ 34.601-90 выделяют следующие стадии и этапы создания автоматизированных систем, в том числе программного обеспечения (табл. 2.3). Данный стандарт ориентирован в большей степени на каскадный характер разработки.

Таблица 2.3. Стадии и этапы создания ПО

Стадии	Этапы разработки
Формирование требований к ПО	Обследование объекта и обоснование необходимости создания ПО
	Формирование требований пользователя к ПО
	Оформление отчета о выполненной работе и заявки на разработку ПО (тактико-технического задания)
Разработка концепции ПО	Изучение объекта
	Проведение необходимых научно-исследовательских работ
	Разработка концепций ПО и выбор варианта, удовлетворяющего требованиям пользователя
	Оформление отчета о выполненной работе
Техническое задание	Разработка и утверждение технического задания на создание ПО
Эскизный проект	Разработка предварительных проектных решений по системе и ее частям
	Разработка документации на ПО и ее части
Технический проект	Разработка проектных решений по системе и ее частям
	Разработка документации на ПО и ее части
	Разработка и оформление документации на поставку изделий для комплектования ПО и (или) технических требований (технических заданий) на их разработку
	Разработка заданий на проектирование в смежных частях проекта объекта автоматизации
Рабочая документация	Разработка рабочей документации на систему и ее части
	Разработка или адаптация программ
Ввод в действие	Подготовка объекта автоматизации к вводу ПО в действие
	Подготовка персонала
	Комплектация ПО поставляемыми изделиями (программными и техническими средствами, программно-техническими комплексами, информационными изделиями)

Более новый ГОСТ Р ИСО/МЭК 12207-2010, соответствующий международному стандарту ISO/IEC 12207:2008, выделяет несколько процессов жизненного цикла: основные, вспомогательные, организационные. Данный стандарт больше ориентирован на итерационный тип разработки: на каждом этапе получается работоспособный продукт с заложенной в него возможностью развития (эволюции). Целью последующих этапов является эволюция полученных на предыдущих этапах результатов. Каждый этап является проектом, содержащим все стадии жизненного цикла, соответствующие каскадной модели (см. табл. 2.3). Итерационный подход

обеспечивает «обозримость» каждого этапа разработки, позволяет эффективнее производить тестирование, предоставляет возможность промежуточного согласования проекта с заказчиком. Итерационная концепция разработки хорошо согласуется с объектно-ориентированным программированием и предоставляет больше возможностей для повторного использования кода.

Кроме организационной модели жизненного цикла программного обеспечения существует ряд других, связанных с принципами проектирования и особенностями разработки, заложенными в языковые конструкции:

– модель шаблонов проектирования;
– функциональная модель;
– объектно-ориентированная модель;
– аспектно-ориентированная модель;
– модель бизнес-логики;
– модель структуры данных.

Многообразие моделей, связанных с разработкой программного обеспечения, обусловлено сложностью программных проектов и необходимостью систематизации на всех этапах жизненного цикла, что позволяет использовать ранее созданный программный код, вести подробное документирование и в результате сократить затраты ресурсов на разработку.

Например, парадигма «модель–вид–контроллер» (Model–View–Controller) позволяет создавать приложения, в которых интерфейсная часть (в частном случае, визуализация графического интерфейса или межпрограммное взаимодействие) минимально зависит от бизнес-логики вычислительного ядра. Такой подход существенно способствует повторному использованию программного кода и переносу, например, в другую операционную систему.

В начале 2000-х гг. появилась *аспектно-ориентированная модель* разработки программного обеспечения, согласно которой сквозная функциональность выделяется в отдельную сущность. В ней используется понятие «аспект» – программный модуль, оформленный в виде функций и(или) классов и реализующий сквозную функциональность.

Актуальной задачей на сегодняшний день является разработка интерфейсной части программного обеспечения, в частности, и информационно-управляющих систем – в целом. Под интерфейсной частью мы понимаем некоторую часть системы (например, программного обеспечения), являющейся подсистемой взаимодействия с другими системами. Особенности интерфейсной части программного обеспечения, её соответ-

ствия распространённым стандартам во многом определяет возможности использования, а соответственно и распространения.

Когда выбраны аппаратная архитектура и средства разработки программного обеспечения, возникает ряд задач: оптимизация кода, отладка, сопровождение, контроль версий, документирование, рефакторинг.

Многие современные интегрированные среды разработки (IDE, Integrated Development Environment), такие как Intell Parallel Studio, Embarcadero RAD Studio, Microsoft Visual Studio, Qt Creator, Eclipse, NetBeans, Xcode содержат практически полный подключаемый или встроенный инструментарий для решения задач на всех стадиях разработки.

Хороший стиль разработки программного обеспечения определяется следующими принципами: максимально возможное исключение повторяющегося кода, полное документирование и комментирование, применение проверенных практикой паттернов и шаблонов, в том числе для выделения и очистки памяти и обработки исключительных ситуаций.

Комментирование и документирование позволяют менять команду разработчиков, поддерживать код по окончании разработки и предоставлять его сторонним разработчикам. Великий русский путешественник и исследователь Н. М. Пржевальский говорил: «у путешественника нет памяти», поэтому регулярно вёл путевые заметки. Разработчик программного обеспечения, подобно путешественнику, памяти не имеет. Если нет надлежащих комментариев и документации, код, написанный пять лет и даже неделю назад, может превратиться в головоломку для автора и потребовать значительного времени и усилий для ее разгадки.

Многие разработчики и их сообщества стараются придерживаться определённых стилей программирования, построенных на едином форматировании кода; осмысленном и синтаксически обоснованном присвоении имён переменным, функциям, классам; комментировании кода. Сам код обычно создаётся с использованием некоторых паттернов и достаточно универсальных библиотек, возможно от сторонних разработчиков. Такой код при некоторых навыках легко читается, и его многие части не требуют отдельного внимания. Принципиальным плюсом этого подхода является возможность сосредоточить внимание на уникальных для данной программы решениях (которые, в большинстве случаев, могут занимать лишь малую часть от всего объёма кода). Однако унификация приводит к недостаточной оптимизации кода под конкретные задачи, а значит и сравнительно невысокому быстродействию. *Рефакторинг* – доработка программного кода до более удобного для восприятия вида без изменения функциональности программы – имеет особое значение.

Оптимизация программ является отдельной задачей разработки программного обеспечения [28]. Касперски К. в книге «Техника оптимизации программ» [73] предлагает выполнять оптимизацию:

– на уровне структур данных и алгоритмов их обработки;

– на уровне планирования потоков машинных команд;

– на уровне компиляции, выбор и сравнение качества оптимизации различными компиляторами;

– работы с периферией (жесткие и лазерные диски, коммуникации, параллельные и последовательные порты, видеоподсистема);

–параллельных вычислений.

Наиболее ощутимые результаты дает оптимизация на уровне структур данных и алгоритмов их обработки [109]. Решение множества практических задач облегчают библиотеки, содержащие наиболее совершенные алгоритмические решения [110, 112].

Существенную помощь в вопросах оптимизации могут оказать *профилировщики* – специальные пакеты программ, предназначенные для анализа программного кода во время выполнения (в частности, VTune, CodeAnalyst, AQtime). Профилировщик помогает выявлять затратные (времени исполнения, потреблению памяти, количеству вызовов) фрагменты кода и принимать решения по оптимизации программ. Задача профилировки сопряжена с задачами *дизассемблирования* и *обратного инжиниринга*, т.е. анализа и разбора машинного кода. Мощным программным пакетом, предназначенным для решения задач обратного *инжиниринга*, является IDA (Interactive DisAssembler).

При разработке программного обеспечения требуются доработка, улучшение, расширение функционала и другая модификация кода. Изменения могут затронуть весь код или некоторую его часть. При этом необходимо *управлять версиями* (при этом требуется особая аккуратность, поскольку риск ошибки разработчика достаточно высок) и выполнять *актуализацию*. Современные средства разработки содержат интегрированные сторонние или собственные системы управления версиями (VCS, Version Control System).

2.6. Оптимизация вычислений путём преобразования форматов

Единицей измерения информации и самым малым объектом хранения и передачи двоичной информации является *бит*. Следующей после бита единицей информации является *байт*, содержащий в своём составе несколько битов. Наиболее распространённая архитектура современных ЭВС ставит в соответствие одному байту восемь упорядоченных (0–7) битов. На языке ассемблера байт обозначается директивой `DB`. Далее в иерархии машинных форматов находятся *слово* (в ассемблере `WORD` или `DW`),

состоящее из шестнадцати бит, или двух байт, и *двойное слово* (в ассемблере `DWORD` или `DD`), состоящее из тридцати двух бит, или четырёх байт. Информация об обозначениях типа данных, диапазона допустимых значений, разрядности для некоторых базовых типов, принятых в языке C++, приведена в табл. 2.4.

Таблица 2.4. Типы данных C++

Тип C++	Псевдоним (Alias)	Разрядность, бит	Диапазон	
			минимум	максимум
bool	bool	8	true или false	
char (int8)	int8	8	– 128	127
unsigned char	unsigned int8	8	0	255
short int	int16	16	– 32768	32767
unsigned short int	unsigned int16	16	0	65535
long int	int32	32	– 2147483648	2147483647
unsigned long int	unsigned int32	32	0	4294967295
—	int64	64	–2E+63	(2E63)–1
—	unsigned int64	64	0	(2E64)–1
float	—	32	3.4E–38	3.4E+38
double	—	64	1.7E–308	1.7E+308
long double	—	80	3.4E–4932	3.4E+4932

Следует учитывать, что различные компиляторы могут по-разному трактовать определения некоторых типов, в зависимости от разрядности аппаратной платформы (8, 16, 32, 64 и т.д.), настройки компилятора и т.д. Поэтому в некоторых случаях целесообразно явно указывать размер и тип, например, вместо `short int`, использовать int16.

Проанализировав табл. 2.4, можно выделить формат чисел с плавающей точкой и целочисленный формат (рис. 2.2).

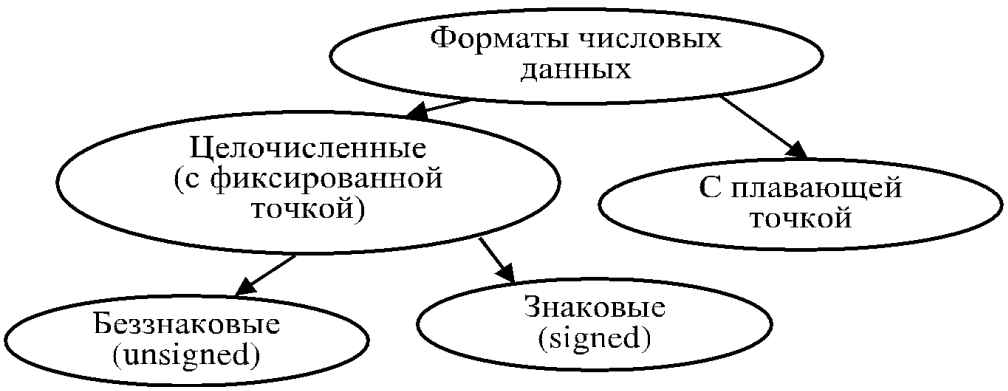


Рис. 2.2. Классификация базовых форматов числовых данных

В каком формате предпочтительно обрабатывать и хранить информацию, в частности, цифровые сигналы? У каждого формата есть свои преимущества и недостатки. Формат с плавающей точкой в некотором смысле можно считать «аналоговым», он обладает широким диапазоном значений, возможностью сохранения дробных величин с высокой точностью. К недостаткам формата относится то, что:

– вычисления выполняются относительно медленно;

– сигналы от большинства датчиков после АЦП переводятся в целочисленный формат;

– большинство «потребителей» цифровых сигналов, в том числе устройства визуализации, воспроизведения звука, требуют данных в целочисленных форматах, для чего необходимо преобразование форматов.

На рис. 2.3 приведены две схемы обработки сигналов (ФТФ – формат с фиксированной точкой; ПТФ – формат с плавающей точкой; АЦП – аналогово-цифровой преобразователь; ЦАП – цифро-аналоговый преобразователь). Схема *а* обеспечивает высокое быстродействие, но накладывает некоторые, порой недопустимые, ограничения на процедуру преобразования; схема *б* позволяет производить самые разнообразные цифровые преобразования сигнала, но существенно снижает быстродействие на этапах обработки сигнала в формате с плавающей точкой и преобразования форматов.

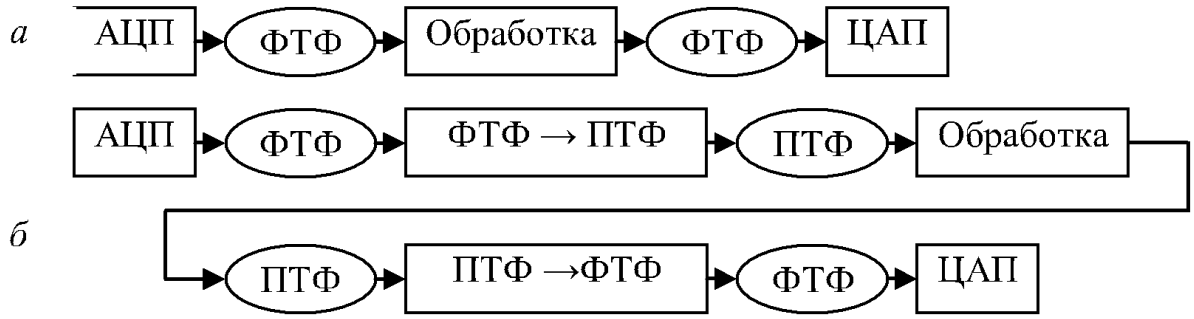


Рис. 2.3. Схемы обработки цифровых сигналов

Возможным решением задачи повышения быстродействия является использование схемы, приведённой на рис. 2.4 (Ф1 – ФТФ, совместимый с ЦАП и АЦП; Ф2 – ФТФ, позволяющий произвести требуемые преобразования). В этой схеме все операции выполняются в целочисленных форматах, а требуемая точность достигается за счёт увеличения числа разрядов, фактически за счёт перехода к формату с фиксированной точкой.

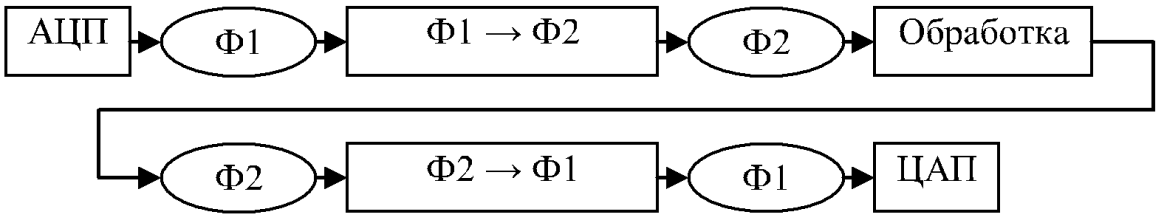


Рис. 2.4. Альтернативная схема цифровых преобразований сигналов

Например, в большинстве цветовых схем на каждый цветовой канал изображения выделяется восемь бит. Сложение двух или более восьмибитных элементов в формате типа `unsigned char` может привести к переполнению расчетных регистров, ячеек памяти. Использование переменных большей разрядности позволяет, гарантированно не вызывая переполнения (до определённого значения суммы), производить сложение: например, формат `unsigned int16` позволяет выполнять сложение до 256 элементов типа `unsigned char`.

В целочисленной арифметике умножение на число, кратное степени двух, может быть выполнено с помощью поразрядного сдвига влево. При операциях над сигналами, каждый дискретный элемент которых представлен в формате `unsigned char`, действия в формате `unsigned int16` могут вызвать переполнение уже при втором умножении, в `unsigned int32` – при четвёртом. Можно заключить, что при использовании целочисленных форматов для операции умножения необходимо ограничивать результаты после выполнения соответствующего числа операций. Практическая реализация такого ограничения зависит от особенностей конкретной программно-аппаратной платформы и способа решения задачи.

Наиболее затратной с точки зрения потребления вычислительных ресурсов является операция деления. В целочисленной арифметике деление на число, кратное степени двух, может быть выполнено с помощью поразрядного сдвига вправо. Заметим, что деление можно с заданной степенью точности реализовать с помощью операций сдвига и сложения. При делении в целочисленной арифметике может возникать существенная потеря точности за счёт отбрасывания дробной части.

Обобщим сказанное в формальной записи. В цифровой обработке преимущественно используется умножение с последующим суммированием, например, свёртка, быстрое преобразование Фурье, фильтр скользящего среднего реализуется на базе умножения с последующим суммированием. Пусть по схеме (рис. 2.4) необходимо в ФТФ выполнить умножение с последующим суммированием N некоторых значений α_i на коэффициенты c_i:

$$S = \sum_{i=0}^{N-1} \alpha_i c_i \ , \qquad (2.1)$$

где α_i – число в ФТФ; c_i – число в ПТФ; S – численный результат суммы в ПТФ.

Для исключения переполнения, когда все $\alpha_i = 0, ..., N-1$, имеют максимальную величину, необходимо ограничить число элементов N и значения коэффициентов c_i.

Ограничить значение N при необходимости можно, разделив сумму (2.1) на:

$$S = \sum_{i=0}^{N-1} \alpha_i c_i \ = \sum_{i=0}^{M-1} \alpha_i c_i \ + \sum_{i=M}^{P-1} \alpha_i c_i \ + \ldots + \sum_{i=T}^{N-1} (\alpha_i c_i) \ . \qquad (2.2)$$

Ограничить значения c_i возможно, например, при помощи нормирования:

$$\overline{c}_i = \frac{c_i}{\sum_{i=0}^{N-1} c_i} \ . \qquad (2.3)$$

Далее, с учётом ФТФ, в котором будет производиться умножение с суммированием, необходимо принять решение: сколько знаков после точки должно быть учтено. Пусть значимое число знаков будет k, тогда коэффициент h возможно вычислить как $h=0x10^k=10_{16}{}^k=16^k$, где $0x$ – префикс 16-ричной системы счисления. Основание степени 16 обеспечивает некоторую избыточность, по сравнению с основанием десятичной системы счисления, повышая точность вычислений. В общем случае при $h=2^k$ (k – целое число) операции умножения и деления на h легко выполнить с помощью сдвига влево и вправо на $\log_2 h$ бит, для $h=16^k$ $\log_2 h=\log_2(2^{4k})$:

– умножение $\overline{c}_i << 4k = \overline{c}_i (0x10^k)$;

– деление $\overline{c}_i << 4k = \overline{c}_i / (0x10^k)$;

где << и >> – операторы поразрядного сдвига влево и вправо соответственно.

Таким образом, получим результирующее выражение для операций умножения с последующим суммированием:

$$\begin{cases} \overline{c}_i^h \approx \overline{c}_i << 4k, \\ h = \ 2 >> 4k \ << 1, \\ S \ = \hat{S} = \left(\sum_{i=0}^{N-1} (a_i \overline{c}_i^h) + h \right) >> 4k. \end{cases} \qquad (2.4)$$

При вычислениях в машинных системах неизбежны ошибки, вызванные дискретностью всех форматов данных в цифровых ЭВС. Причиной ошибок при дискретных вычислениях является необходимость постоянного округления промежуточных и конечных результатов.

К ошибкам приводит и то, что в форматах для записи чисел с плавающей точкой длины мантиссы и экспоненты постоянны. Числа с меньшим значением экспоненты имеют меньший шаг дискретизации. Вычисления с одновременным использованием чисел разного порядка могут вызвать существенные ошибки.

Принято различать *случайные* и *нарастающие ошибки* (рис. 2.5). Первые компенсируют друг друга и обычно их влияние на результаты вычислений не столь существенно. В некоторых случаях случайные ошибки можно существенно уменьшить при помощи повторных вычислений с последующей обработкой, например, аппроксимацией наилучшего значения.

Нарастающие ошибки приводят к регулярному искажению результатов и не могут быть устранены повторными вычислениями, для борьбы с ними необходимо применять, например, алгоритм Кэхэна.

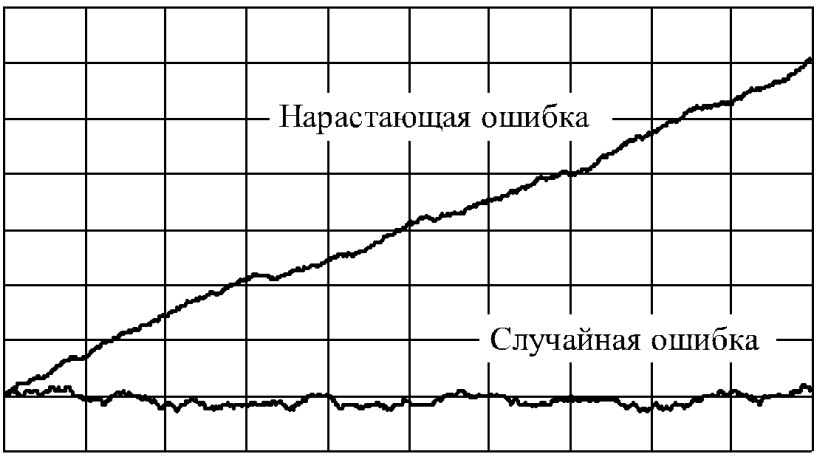


Рис. 2.5. Случайная и нарастающая ошибки

В листинге 2.3 приведён программный код, генерирующий нарастающие и случайные ошибки, при последовательном многократном сложении `err+=dlt` получается значение, отличное от результата единовременного вычисления `err-=dlt*float(step)` (с умножением). При малых значениях `step` преобладает случайная ошибка, нарастающая отсутствует, с увеличением `step` влияние нарастающей ошибки увеличивается, и при достаточно больших `step` она значительно превалирует над случайной. График (рис. 2.5) сгенерирован с помощью программы (см. листинг 2.3), необходимо обратить внимание, что некоторые компиляторы способны «разворачивать» подобный код и оптимизировать, поэтому результаты

расчётов могут различаться для различных компиляторов при включённой опции оптимизации.

Листинг 2.3. Генератор ошибок округления (C++)

```
//=================================================================
float dlt;         // величина приращения
float err=0.0;     // ошибка
int j, i;          // счётчики
int step=150;      // число шагов последовательного сложения
randomize();
for (j=0; j<1000; j++) {
  dlt=(float)RAND_MAX+log(1.0+((float)rand()/(float)RAND_MAX));
  // последовательное сложение
  for (i=0; i<step; i++) {
    err+=dlt;
  }
  // объединённое вычитание с умножением
  err-=dlt*(float)step;
}
//=================================================================
```

При разработке программного обеспечения для снижения влияния ошибок округления на результаты вычислений необходимо учитывать вычислительные особенности платформы и языка реализации.

2.7. Практическое применение оптимизации вычислений на базе преобразования форматов

Оптимизированный вариант преобразования данных из цветового пространства RGB в YCbCr (вариация $Y_{709}CbCr$) в соответствии с выражением (1.11), построен на базе рассмотренных ранее принципов целочисленных вычислений (листинг 2.4). Для перевода в ФТФ использовался множитель $k = 2^{20} = 1\,048\,576$, соответственно для округления к ближайшему целому можно использовать добавление величины $0.5k$.

Листинг 2.4. Преобразование значения из цветового пространства RGB в YCbCr (C++)

```
//=================================================================
ColorSpace x;
...
int R, G, B;
R=(int)x.byteLo;
G=(int)x.byteAv;
B=(int)x.byteHi;
const int h=(2^20)>>1; //  h=524288;
// компонент (Y)
x.byteLo=(uchar)(abs(191889*R+643826*G+65012*B+16777216+h))>>20);
// компонент (Cb)
x.byteAv=(uchar)(abs(-105906*R-
354419*G+460325*B+134217728+h)>>20);
```

```
// компонент (Cr)
x.byteHi=(uchar)(abs(460325*R-418382*G-41943*B+134217728+h)>>20);
...
//================================================================
```

Сравнения преобразований на базе вычислений ПТФ и ФТФ (рис. 2.6), показали, что время операций было сокращено более чем в восемнадцать раз. Подобное повышение скорости вычислений иллюстрирует эффективность перехода (при возможности) к ФТФ даже при наличии математического сопроцессора.

Рис. 2.6. Скорость вычисления преобразования цветовых пространств, изображение размером 4000×2000

Обратные преобразования из YCbCr в RGB можно выполнить в соответствии с выражением (1.12), реализованным в ФТФ (листинг 2.5, для уменьшения искажений вследствие округления к целому применены поправочные коэффициенты).

Листинг 2.5. Преобразование значения из цветового пространства YCbCr в RGB (C++)

```
//================================================================
ColorSpace x;
...
int Y, Cb, Cr;
Y=(int)x.byteLo;
Cb=(int)x.byteAv;
Cr=(int)x.byteHi;
// поправочные коэффициенты
const int h1=354615;
const int h2=614460;
// компонент (R)
x.byteLo=(uchar)(abs(1220689*Y-1152*Cb+1879436*Cr-259951428-
h1))>>20;
// компонент (G)
x.byteAv=(uchar)(abs(1220689*Y-223447*Cb-560263*Cr+80783763))>>20;
// компонент (B)
x.byteHi=(uchar)(abs(1220689*Y+2216249*Cb+1035*Cr-303343600-
h2))>>20;
...
//================================================================
```

Следует отметить, что в программах (листинги 1.3, 2.4, 2.5) в результате целочисленных округлений преобразования становятся нелинейными, следствие нелинейности – искажения и частичная потеря информации, визуально незаметная, но, возможно, критичная для технических приложений. При множественных взаимных преобразованиях искажения могут накапливаться. Исключить или уменьшить их можно, изменив алгоритм и добавив некоторые условия, это приведет к замедлению вычислений (особенно заметному для циклов попиксельной обработки изображения). Поэтому часто применяют «оперативные методы коррекции», в частности, для «повышения однозначности» взаимного преобразования листинге 2.5 введены поправочные коэффициенты (`h1` и `h2`).

Приведённые в данном труде вычисления машинного времени производились на компьютере, построенном на базе процессора Core2Duo E8500, с эффективной тактовой частотой 3,16 ГГц, частотой процессорной шины 1.333 ГГц, кэш-памятью второго уровня размером 6 МБ, в среде OS Windows XP и Windows-7.

2.8. Оптимизация вычислительных циклов

Способ организации выделенного адресного пространства может существенно повлиять на эффективность вычислений [73]. В листинге 2.6 сравниваются способы перебора двумерного массива. Результаты работы программы для массива `data` размером 8192×8192 приведены на рис. 2.7. Очевидно, что скорость цикла перебора по столбцам, вложенного в цикл перебора по строкам, примерно в два раза выше, чем у другой пары вложенных циклов. Здесь надо отметить, что понятия «строка» и «столбец» достаточно условны, ведь память ЭВС линейна. В соответствии с последовательностью выделения памяти (листинг 1.2) соседние элементы строки расположены по соседним адресам в физической памяти, а элементы столбцов удалены друг от друга. Следовательно, при проходе по строке более эффективно используются преимущества быстродействующей кэш-памяти, логика предсказания и меньше обращений к более медленной оперативной памяти.

Заметим, что быстрее заполнять фрагменты памяти возможно с помощью функций стандартной библиотеки С, таких как `memcpy`, `memset`; или C++: например, инициализацией вектора (`std::vector`) при объявлении. При решении более сложных задач, выбрав оптимальную последовательность перебора, можно существенно повысить быстродействие исполняемой программы.

Листинг 2.6. Сравнение способов перебора элементов двумерного массива (C++)

```
//=============================================================
// заполнение двумерного массива изображения data
// значением, содержащимся в temp
...
UColorSpace temp;
...
// проход по столбцам
for (uint j=0; j<iWidth; j++) {
  // проход по строкам
  for (uint i=0; i<iHeight; i++) {
    temp.numb=(i+j);
    data[i][j]=temp.rgb;
  }
}
...
// проход по строкам
for (uint i=0; i<iHeight; i++) {
  // проход по столбцам
  for (uint j=0; j<iWidth; j++) {
    temp.numb=(i+j);
    data[i][j]=temp.rgb;
  }
}
//=============================================================
```

Рис. 2.7. Время выполнения теста:
способ перебора элементов двумерного массива

Оптимизация программы может обеспечить существенное повышение скорости обработки данных.

3. Устранение искажений, коррекция и калибровка изображений

Искажения присущи практически всем цифровым сигналам, они могут существенно осложнить обработку и анализ изображений, а также восприятие их человеком. Поэтому необходимо бороться с различными видами искажений на всех этапах получения и преобразования изображения. Причиной возникновения искажений может быть:

– несовершенство приёмных устройств, например, неидеальная форма линз оптических камер, дефекты ПЗС-матриц и т.д.;

– аддитивное и мультипликативное добавление шумов при передаче по линиям радиосвязи или существенное затухание сигнала в линиях связи;

– искажение сигнала во время обработки, например, при масштабировании и повороте изображения.

Искажения проявляются преимущественно в частотной или в пространственной области представления сигнала или одновременно в обеих областях. Исходя из специфики проявления искажений, с учётом причин их возникновения, следует выбирать способы устранения. В подавляющем большинстве случаев полностью исключить искажения невозможно, да и на практике достаточно частичного их устранения.

С искажениями следует бороться на всех этапах получения и преобразования сигнала, и в первую очередь необходимо понять причины их возникновения. Вследствие многообразия видов искажений важнейший этап борьбы с ними – выбор адекватной математических моделей системы формирования изображений и объекта изображения. Задачу устранения искажений можно отнести к классу обратных задач цифровой обработки сигналов.

3.1. Управление диапазоном значений сигналов

В ЭВС преобразования значений элементов, образующих сигнал, могут приводить к переполнениям, потере точности, смещению плотности распределения величины сигнала, следствием этого могут быть искажения. Результаты преобразования сигналов может существенно искажать использование различных числовых форматов (см. табл. 2.4).

Округление значений при переходе от формата с плавающей точкой к формату с фиксированной точкой. Наиболее распространено округление:

– в меньшую сторону (получение наибольшего целого, не превышающего x):

$$w[x] = \lfloor f[x] \rfloor; \qquad (3.1)$$

– в большую сторону (получение наименьшего целого, не меньшего x):

$$w[x] = \lceil f[x] \rceil; \tag{3.2}$$

– до ближайшего целого:

$$w[x] = \lfloor f[x] + 0.5 \rfloor \text{ или } w[x] = \lceil f[x] - 0.5 \rceil. \tag{3.3}$$

Переход в беззнаковую область. При преобразованиях сигналов в форматах со знаком с плавающей и/или с фиксированной точкой возможно возникновение как положительных, так и отрицательных значений. При переходе к беззнаковым форматам можно воспользоваться добавлением константы (например, абсолютной величины минимального отрицательного значения) или определением модуля (рис. 3.1):

$$\begin{cases} w[x] = f[x] + \text{const} \\ \text{const} \geq f[x]_{\min} \end{cases} \tag{3.4}$$

$$w[x] = |f[x]|. \tag{3.5}$$

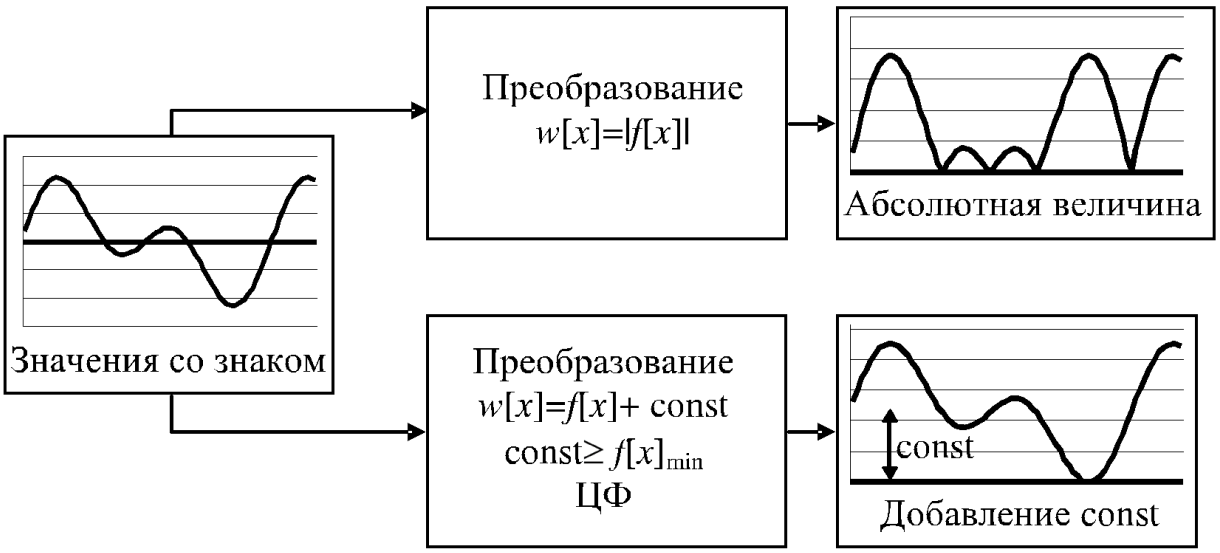


Рис. 3.1. Преобразование в беззнаковый формат

Очевидно, что полученные результаты различны, но каждый из описанных подходов может быть полезен в определённой ситуации. Например, при визуализации изображений, отсчёты которых (со значительной амплитудой – как положительной, так и отрицательной) важно различать, может быть полезным преобразование в беззнаковый формат путём вычисления абсолютной величины. С другой стороны, при определении абсолютной величины исходная форма сигнала утрачивается, это может быть недопустимо, особенно если требуется дальнейшая обработка.

Расширение и нормирование динамического диапазона сигнала часто необходимо выполнять в ходе более сложных преобразований или как самостоятельную задачу ЦОС. Примером может быть HDR (High Dynamic Range) – способ расширения динамического диапазона, применяемый для обработки фото- и видеоизображений. Один из способов реализации HDR

предполагает совместное использование нескольких образов одного объекта, при этом каждый образ наиболее эффективно передаёт области изображения с некоторым выбранным диапазоном значений интенсивности светоизлучения (отражённого или собственного). HDR-технологию применяют в обработке разнообразных снимков – от медицинских до космических, 3D-моделировании, телевидении высокой чёткости (High-Definition Television) и др.

На базе расширения и нормирования динамического диапазона реализуют так называемую *гамма-коррекцию* – изменение динамического диапазона сигнала с помощью степенной функции, применительно к графической информации она позволяет:

– эффективно использовать устройства визуализации изображений (мониторы, проекторы и др.) – яркость элементов визуализации практически всех устройств не линейно зависит от уровня сигнала, а пропорциональна некоторой степенной функции (с показателем γ);

– корректировать изображения для печати – вследствие особенностей печатающих устройств и различных расходных материалов может требоваться значительная предварительная обработка изображений (преобразование цветового пространства, гамма-коррекция);

– повышать контрастность некоторой области изображения – это может быть необходимо при обработке различных графических объектов, например, медицинских и космических снимков.

На рис. 3.2 приведён пример нелинейной трансформации динамического диапазона.

Когда результаты прямого и обратного преобразования выходят за диапазон допустимых значений или не могут быть сохранены с приемлемой точностью из-за дискретного разрешения формата, взаимно-однозначное обратное преобразование невозможно, это практически всегда приводит к частичной потере информации.

Рассмотрим некоторые особенности функции преобразования (рис. 3.2) при различных значениях предела отношения:

– расширение динамического диапазона:

$$\lim_{\Delta f(x)\to 0} \frac{\Delta w(x)}{\Delta f(x)} > 1, \tag{3.6}$$

– ширина диапазона неизменна:

$$\lim_{\Delta f(x)\to 0} \frac{\Delta w(x)}{\Delta f(x)} = 1, \tag{3.7}$$

– сужение динамического диапазона:

$$0 < \lim_{\Delta f(x)\to 0} \frac{\Delta w(x)}{\Delta f(x)} < 1, \tag{3.8}$$

– сужение и инверсия динамического диапазона:

$$-1 < \lim_{\Delta f(x)\to 0} \frac{\Delta w(x)}{\Delta f(x)} < 0, \tag{3.9}$$

– инверсия динамического диапазона при неизменной ширине:

$$\lim_{\Delta f(x)\to 0} \frac{\Delta w(x)}{\Delta f(x)} = -1, \tag{3.10}$$

– расширение и инверсия динамического диапазона:

$$\lim_{\Delta f(x)\to 0} \frac{\Delta w(x)}{\Delta f(x)} < -1. \tag{3.11}$$

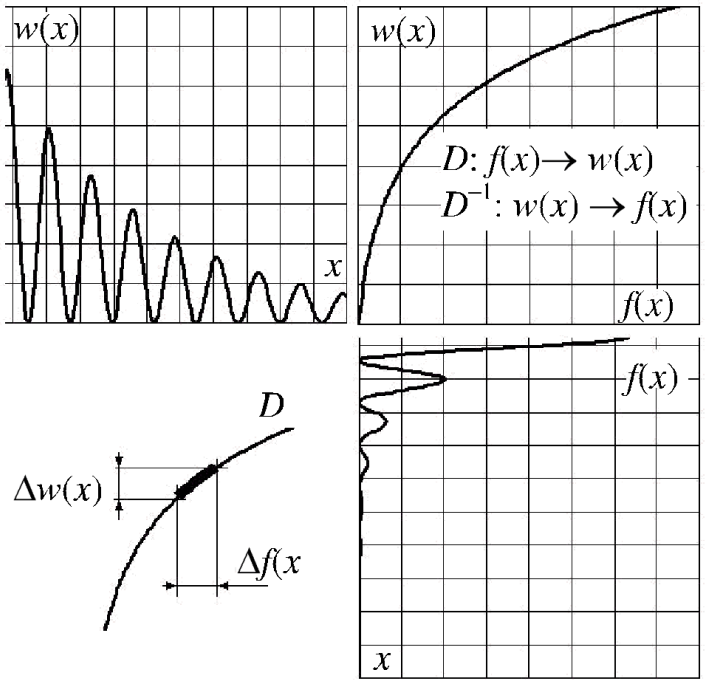


Рис. 3.2. Преобразование динамического диапазона (гамма-коррекция)

Нормирование динамического диапазона выполняется для того, чтобы значения сигнала не превышали допустимых, часто процедуры расширения или сужения диапазона совмещают с нормированием, исходя из требований к результату синтеза функции преобразования.

Согласно *рекомендации* ITU-R BT.709 [154], в цветовом пространстве RGB для визуализации графической информации на электронных устройствах отображения (дисплеях, проекторах и пр.) гамма-коррекция выполняется в соответствии с выражением:

$$\begin{cases} R,G,B < 0.018, \\ R' = 4.5R, \\ G' = 4.5G, \\ B' = 4.5B \end{cases} \quad \text{и} \quad \begin{cases} R,G,B \geq 0.018, \\ R' = 1.099R^{0.45} - 0.099, \\ G' = 1.099G^{0.45} - 0.099, \\ B' = 1.099B^{0.45} - 0.099, \end{cases} \tag{3.12}$$

где R,G,B – исходные значения цветовых составляющих, заданные в диапазоне 0–1, R',G',B' – результаты гамма-коррекции в диапазоне 0–1.

3.2. Масштабирование и трансформация многомерных сигналов

Большинство цифровых сигналов требует различных преобразований, под которыми в общем случае имеются в виду отображения сигналов. Рассмотрим некоторые виды геометрических преобразований.

Наиболее простым видом преобразования является сдвиг, т.е. изменение положения некоторого образа, более сложным – поворот. Сдвиг сигнала $f(x_1, x_2, ..., x_n)$ можно осуществить добавлением к координатам некоторой величины (смещения) $\Delta_1, \Delta_2, ..., \Delta_n$:

$$f(x_1 + \Delta_1, x_2 + \Delta_2, ..., x_n + \Delta_n) = f(x_1', x_2', ..., x_n') . \tag{3.13}$$

Операция сдвига может быть востребована, например, при центрировании некоторого образа. При сдвиге происходит только смещение сигнала, его форма неизменна, если не было выхода за область определения сигнала.

Операция поворота (рис. 3.3) осуществляется путем преобразования координат, в результате умножения координат на матрицу поворота (поворачивающую матрицу). Например, поворот координат в двумерном пространстве осуществляется с помощью выражения:

$$\begin{bmatrix} x_1' \\ x_2' \end{bmatrix} = \begin{bmatrix} \cos\alpha & -\sin\alpha \\ -\sin\alpha & \cos\alpha \end{bmatrix} \cdot \begin{bmatrix} x_1 \\ x_2 \end{bmatrix}, \tag{3.14}$$

где x_1, x_2 – исходные, а x_1', x_2' – результирующие координаты; α – угол поворота.

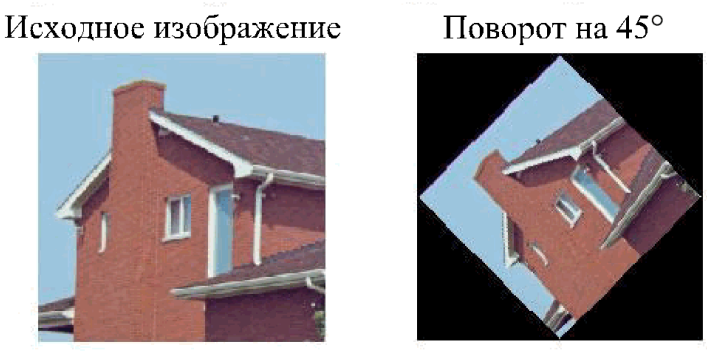


Рис. 3.3. Поворот изображения (с масштабированием)

В трёхмерном пространстве обычно используют три поворачивающие матрицы. Следует отметить, что при визуализации результатов трёхмерного моделирования зачастую выполняется поворот не изображения, а *наблюдателя* (или камеры) – некоторой точки в виртуальном пространстве, относительно которой моделируется вся сцена. В этом случае применяют характеристику «угол Эйлера», или ее интерпретацию для данного виртуального пространства. Углы Эйлера описывают поворот абсолютно твёрдого тела в пространстве:

– вращение вокруг оси X (α – угол крена):

$$R_x = \begin{pmatrix} 1 & 0 & 0 \\ 0 & \cos\alpha & -\sin\alpha \\ 0 & \sin\alpha & \cos\alpha \end{pmatrix}; \qquad (3.15)$$

– вращение вокруг оси Y (β – угол тангажа):

$$R_y = \begin{pmatrix} \cos\beta & 0 & -\sin\beta \\ 0 & 1 & 0 \\ -\sin\beta & 0 & \cos\beta \end{pmatrix}; \qquad (3.16)$$

– вращение вокруг оси Z (γ – угол рыскания):

$$R_z = \begin{pmatrix} \cos\gamma & -\sin\gamma & 0 \\ \sin\gamma & \cos\gamma & 0 \\ 0 & 0 & 1 \end{pmatrix}. \qquad (3.17)$$

При повороте, как и при некоторых других преобразованиях, вследствие дискретности цифрового сигнала, значения исходных координат могут не попасть точно (в целочисленном виде) в значения новых координат. Кроме того, сигнал может быть растянут или сжат, поэтому в большинстве случаев фактические значения необходимо пересчитывать. Исходя из желаемых результатов и возможностей следует выбрать подход к пересчёту значений сигнала при преобразовании координат. Во многих случаях, например для графических изображений, хороший результат даёт аппроксимация (рис. 3.4).

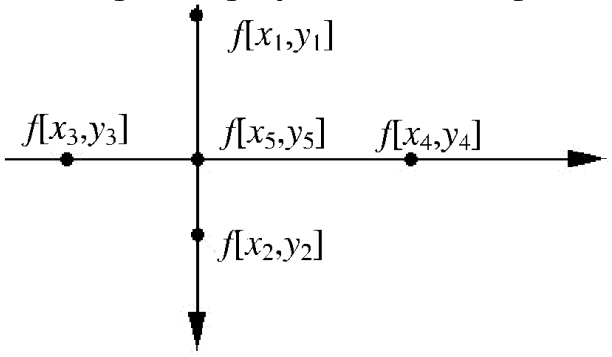


Рис. 3.4. Аппроксимация значения сигнала

Значение сигнала $f[x_5, y_5]$ в точке (x_5, y_5) аппроксимируется относительно точек (x_1, y_1), (x_2, y_2), (x_3, y_3) и (x_4, y_4):

$$f[x_5, y_5] = \frac{1}{2}\left[\frac{(x_5 - x_3) f[x_4, y_4] + (x_4 - x_5) f[x_3, y_3]}{x_4 - x_3} + \right.$$

$$\left. + \frac{(y_5 - y_1) f[x_2, y_2] + (y_2 - y_5) f[x_1, y_1]}{y_2 - x_1}\right]. \qquad (3.18)$$

Достаточно распространено *аффинное* преобразование геометрических цифровых сигналов, т.е. линейное отображение на себя, при котором прямые отображаются в прямые:

$$f : R^n \to R^n . \qquad (3.19)$$

Например, аффинные преобразования для двумерного пространства могут быть описаны следующей системой уравнений:

$$\begin{cases} x' = m_{11} x + m_{12} y + m_{13}, \\ y' = m_{21} x + m_{22} y + m_{23}, \end{cases} \qquad (3.20)$$

где x, y – исходные координаты точки; x', y' – координаты, полученные в результате аффинных преобразований; m_{ij} – коэффициенты аффинных преобразований: $\mathbf{M} = \begin{vmatrix} m_{11} & m_{12} & m_{13} \\ m_{21} & m_{22} & m_{23} \end{vmatrix}$. Коэффициенты матрицы $\mathbf{M}$ определяют результат аффинных преобразований и соответственно – отображения исходного сигнала. Очевидно, что сдвиг, поворот и масштабирование являются частными случаями аффинных преобразований.

Весьма распространена задача коррекции геометрических искажений вызванных, например, несовершенством передающего тракта оптоэлектронного приёмника или особенностями положения объекта, регистрируемого фото- или видеоприёмником.

3.3. Классификация шума

К шумам обычно относят некоторую часть сигнала, в той или иной степени препятствующую восприятию (распознаванию, анализу, выделению) полезной информации. В некоторых случаях шум сам по себе является полезной информацией, например, исследование реликтового излучения Вселенной позволяет оценивать её параметры (возраст, температуру и т.д.). Радиостанции, излучая сигналы сложной модуляции, порождают гармонические компоненты, которые, проходя на частоты вещания других радиостанций, могут восприниматься как шумы. Таким образом, один и тот же сигнал в некоторых случаях может быть отнесен к шумам, а в других – к полезной информации.

Шумы присутствуют практически во всех цифровых сигналах. Возможно выделить характерные признаки, отличающие различные виды шума.

1. Плотность распределения амплитуды, согласно которой шум может считаться гауссовым, максвелловским, равномерным и пр.

2. Спектральная плотность шума определяется характерной формой спектра, т.е. частотным составом шума. Наиболее распространённые виды шума: белый – со сплошным равномерным спектром в выделенном диапазоне частот, а также его разновидности (розовый, красный, серый и т.д.); шумы, имеющие линейчатый спектр. Для полупроводниковой аппаратуры характерны фликкер-шум (имеет розовый спектр) и дробовой шум.

3. Пространственная и временная плотность распределения шума. Например, пространственная неоднородность плотности шумов может возникнуть в результате синтеза (путём склейки) нескольких сигналов, полученных при различных условиях.

4. Регулярные и нерегулярные (случайные) шумы: например, регулярность пространственных шумов в цифровых фотоснимках может быть вызвана дефектами оптики или ПЗС-матрицы (такие шумы называют локальными), периодические шумы во времени в той же ПЗС-матрице могут быть вызваны периодически возникающей электромагнитной помехой. Спрогнозировать появление нерегулярных или случайных шумов достаточно сложно.

5. Шумы естественного (тепловой шум, шум атмосферных электромагнитных явлений и т.д.) и антропогенного происхождения (шумы, вызванные работой электроустановок, загрязнением атмосферы и т.д.).

6. Стационарные и нестационарные шумы, так же как и стационарные и нестационарные случайные процессы, характеризуется изменением вероятностных характеристик с течением времени [106].

Для снижения уровня шума на этапе аналоговой обработки и аналого-цифровых преобразований применяют малошумящие полупроводниковые схемотехнические элементы, охлаждение токоведущих и оптических элементов, стабилизацию питающих напряжений, экранирование и ряд других мер, которые можно отнести к аппаратным решениям. Программные решения можно рассматривать как другой вид шумоподавления.

3.4. Фильтры шума

Вследствие многообразия причин возникновения шума не существует универсальных фильтров, детектирующих и подавляющих все виды шумов, поэтому выделяют несколько принципиально различных подходов к программному или аппаратно-программному снижению уровня шума.

Многие фильтры шума работают по принципу подавления области частот, в которой шум наиболее проявляется. Недостатком частотной фильтрации может быть потеря значимой информации. Например, при

фильтрации случайных шумов в двумерном изображении с помощью выбранной разновидности однородного фильтра (скользящего среднего) на базе операции свёртки с функцией рассеяния точки (ФРТ) 3×3 :

$$h[x,y]=\begin{matrix}1/9 & 1/9 & 1/9\\ 1/9 & 1/9 & 1/9\\ 1/9 & 1/9 & 1/9\end{matrix} \qquad (3.21)$$

или

$$h[x,y]=\begin{matrix}0 & 1/5 & 0\\ 1/5 & 1/5 & 1/5\\ 0 & 1/5 & 0\end{matrix} \qquad (3.22)$$

вследствие усреднения значений и подавления высокочастотных составляющих спектра контрастные границы различных объектов на изображении размываются. Поэтому требуется определить допустимые потери полезной информации, т.е. выявить диапазон частот, подавление которых уменьшит шум, не вызвав существенного искажения полезного сигнала.

Следует отметить, что подавление высоких частот в спектре сигнала с помощью операторных преобразований может привести к появлению низкочастотного шума, который в некоторых случаях визуально более заметен, чем высокочастотный.

В работе [140] Де Хаан предлагает оригинальный свёрточный фильтр, в котором в качестве значения центрального элемента используется взвешенная сумма пикселов, следующих через один.

Основанный на избыточности информации метод снижения шумов может дать очень хорошие результаты, при этом существенной проблемой является значительное время вычислений и увеличение необходимых объёмов информации.

В соответствии с теоремой Ляпунова[5] [97] для подчиняющихся одному закону распределения независимых случайных величин $X_1, X_2, ..., X_n$ с математическим ожиданием m и дисперсией σ^2, при неограниченном увеличении n ($n\to\infty$) закон распределения суммы

$$Y_n=\frac{\sum_{i=1}^{n}X_i}{\sigma\sqrt{n}} \qquad (3.23)$$

мало отличается от нормального.

[5] Александр Михайлович Ляпунов (25 мая 1857, Ярославль – 3 ноября 1918, Одесса) – математик и механик, член-корреспондент Петербургской Академии наук (1900); академик (1901). Ученик П. Л. Чебышева.

Практическим следствием теоремы Ляпунова является то, что сложение случайных некоррелированных шумов не приводит к увеличению суммарного шума, имеющего нормальную плотность распределения. Теорема остаётся справедливой и для суммы случайных некоррелированных величин с неодинаковыми законами распределения при некоторых дополнительных условиях, которые, как правило, выполняются на практике для рассматриваемых случайных величин. Как показывает опыт, сумму порядка десяти слагаемых можно считать нормально распределённой [97]. В практических задачах цифровой обработки сигналов для сокращения вычислительных затрат по возможности ограничиваются тремя-пятью слагаемыми.

Хорошим примером является передача сигнала малой мощности в радиоэфире с шумами, такая задача может возникать при передаче данных со спутника. При достаточно протяжённых радиотрассах и практически постоянной передаче различных данных (телеметрических, космических снимков и т.д.), мощность спутниковых радиопередатчиков существенно ограничена мощностью доступного источника питания. Способ распознавания слабого сигнала на фоне шума – многократная передача повторяющихся фрагментов (пакетов), накопление пакетов со сложением на принимающей стороне. Если математическое ожидание шума в каждом пакете близко к нулю (или другой константе), то математическое ожидание суммарного шума также будет близко к нулю (или константе), причем чем больше пакетов, тем ближе. При этом математическое ожидание суммы пакетов будет стремиться к незашумлённому значению сигнала, умноженному на число пакетов. При некотором числе принятых и сложенных пакетов уровень сигнала превысит уровень шума на величину, достаточную для распознавания. Непременным условием эффективной фильтрации является отсутствие корреляции между шумами различных пакетов. Реализация рассмотренного способа проиллюстрирована в листинге 3.1, а результаты её выполнения – на рис. 3.5. Теоретически фильтрация за счёт накопления позволяет выделять сигнал с любым сколь угодно малым отношением сигнал/шум.

Листинг 3.1. Подавление случайного шума за счёт накопления пакетов (MATLAB)

```
%==================================================================
clear all;        % удаление переменных
close all;        % закрытие всех окон
clc;              % очистка командного окна

% определение исходных данных
size=100;    % число отсчётов сигнала и шума
nnoise=100; % число накопленных пакетов
x=4*pi;      % задание сигнала
data=sin(-x:2*x/(size-1):x);
```

```
a=4;          % амплитуда шума
noise=zeros(size,nnoise);     % начальная инициализация матрицы
% формирование матрицы пакетов,
% образованной векторами суммы (сигнал + шум)
for i=1:nnoise
  noise(:,i)=2*a*rand(100,1)+data(:)-a;
end
% накопление пакетов
result=zeros(size,nnoise); % начальная инициализация
result(:,1)=noise(:,1);
% непосредственно цикл суммирования зашумлённых пакетов
% для наглядности фильтрации пакеты суммируются в матрицу,
% каждый последующий вектор матрицы
% образован большим числом пакетов
for i=2:nnoise
  result(:,i)=result(:,i-1)+noise(:,i);
end
% нормирование значений
for i=2:nnoise
  result(:,i)=result(:,i)/i;
end
% визуализация результата
% результат в 3D
figure;
surf(result);
% выборочные результаты 2D
figure;
subplot(2,2,1); plot(result(:,1));
title('Пакет (сигнал + шум)');
subplot(2,2,4); plot(result(:,2));
title('Усреднение по сумме 2 пакетов');
subplot(2,2,2); plot(result(:,10));
title('Усреднение по сумме 10 пакетов');
subplot(2,2,3); plot(result(:,500));
title('Усреднение по сумме 500 пакетов');
%=================================================================
```

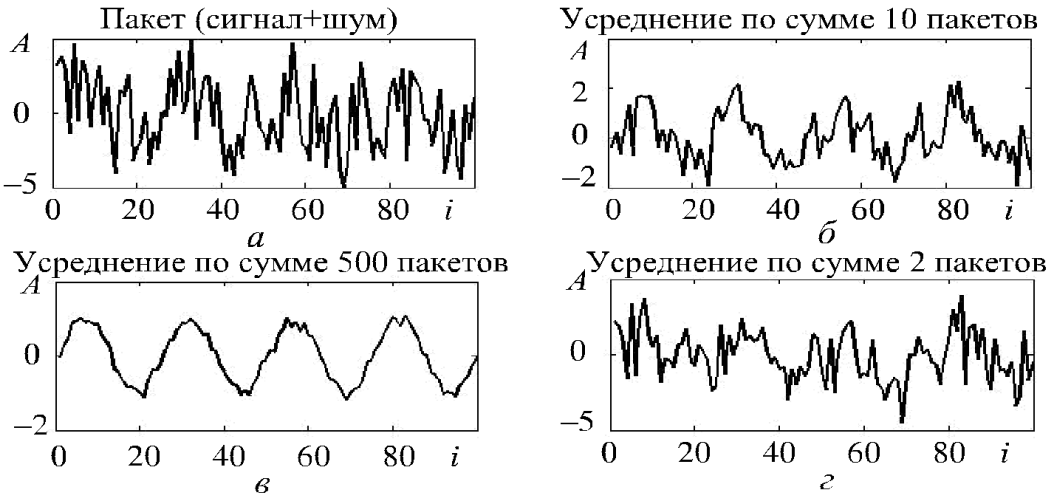


Рис. 3.5. Подавление случайного шума за счёт накопления пакетов

Подход с накоплением информации можно применять для фильтрации динамических изображений. В большинстве случаев такие изображения состоят из последовательности кадров $...f(X,Y,t-1), f(X,Y,t), f(X,Y,t+1)...$, при этом соседние кадры мало отличаются друг от друга, исключением могут быть кадры, соответствующие резкой смене сцены. Данную особенность динамических изображений можно использовать для фильтрации случайных шумов (обычно достаточно 3–5 кадров последовательности). Преодолеть возникающее при такой фильтрации размытие границ движущихся объектов отчасти позволяет использование адаптивных фильтров и/или автоматизированный анализ распознавания движения.

Фильтрация за счёт накопления информации неэффективна для подавления регулярных шумов, поскольку они коррелируют между собой. Для подавления регулярного шума можно использовать способы, основанные на получении тестовых (калибровочных) сигналов, а также применять адаптивные фильтры. Например, для определения в радиоэфире шумов регулярной природы (гармонические составляющие радиосигналов, помехи от электрооборудования и т.д.) можно производить приём сигнала в период, когда радиопередатчик, связанный с данной частотой, соблюдает радиомолчание или посылает заранее известный (калибровочный, тестовый) сигнал – таким образом возможно идентифицировать регулярные шумы. В некоторых устройствах распознавания голоса фильтрацию шумов производят на основе данных, полученных в промежутках между словами. Выявление собственного регулярного шума фото- и видеоаппаратуры можно производить с помощью специальных тестовых снимков.

3.5. Стохастический резонанс

Термин *стохастический резонанс* впервые использован в работах 1981–1982 гг. [132, 133] при изучении периодичности ледниковых периодов и их моделировании с помощью бистабильного осциллятора. Явление стохастического резонанса свойственно мультистабильным (обычно метастабильных областей немного – две-три) и нелинейным системам. В работах [132, 133] показано, что по периодичности с ледниковыми периодами совпадают колебания эксцентриситета Земли. Модельные расчёты показали, что сами по себе они не могут вызвать столь существенных изменений климата, но при введении дополнительной «случайной» силы были получены условия, достаточные для преодоления потенциального барьера (точки бифуркации) и перевода климатической системы в состояние ледникового периода и выхода из него. Последующие исследования описанного явления позволили выявить его наличие в процессах, происходящих в самых различных системах. В 1983 г. стохастический резонанс был исследован в триггере Шмидта [144], а в 1988 г. – в кольцевом лазере [160], в 1991–1994 гг. –

в магнитных [146, 148], биологических и прочих системах. Есть основание полагать, что биологические системы в ходе эволюции приспособились использовать неустранимый внутренний и внешний шум для выделения полезного сигнала [142, 156, 168]. Явление стохастического резонанса в различных системах обобщено в работе [8], в частности, рассмотрены особенности синхронизации фазы, поведение ансамблей систем, которым оно свойственно, а также информационные и энтропийные характеристики резонанса.

Ряд исследований показал, что многим динамическим системам свойственна *точка бифуркации* (от лат. *bifurcus* – раздвоение, вилка), т.е. некоторая пограничная область, разделяющая относительно устойчивые состояния с различным уровнем упорядоченности. В частности, точки бифуркации есть у различных биологических систем [81]. Бифуркация свойственна так называемому *нелинейному мышлению* [86, 89], обусловленному структурой нейронной сети, содержащей обратные связи. Социальным процессам также свойственна нелинейная динамика с периодическими переходами в относительно стабильные состояния. В проведенных Rensselaer Polytechnic Institute (http://scnarc.rpi.edu/) исследованиях социальных процессов на основе модели, построенной по принципу сетевой структуры, выявлено: когда лишь десятой частью общества овладевает некая идея, наступает переломный момент в поведении всего общества и под влиянием этой части оно переходит в некоторое новое состояние [141].

Замечательным примером бифуркации является развитие массовых средств телекоммуникаций (проводная телефония, Интернет, мобильная связь и др.). Новые телекоммуникационные системы интегрируются в социум, сначала являясь достоянием лишь малой части общества, но, стремительно развиваясь, переходят на новый, массовый, уровень. При этом наблюдаются изменения как в социуме, например, появление новых возможностей для общения, трудовых вакансий, инфраструктур и т.д., так и в системе технического оснащения и данных, спрос рождает предложение, а массовый спрос рождает лавину предложений. Таким образом системы переходят на новый информационно-организационный уровень, т.е. меняется упорядоченность системы. Скачок социальной системы на новый информационный уровень стал очевиден с появлением сети Интернет, которая обеспечивает доступ практически к любой информации, наработанной человечеством, стирая границы между слоями общества, языковые преграды и расстояния.

Бифуркация непосредственно связана с уединёнными волнами – солитонами. Бифуркации солитонов впервые были обнаружены в численных экспериментах Лонге–Хиггинса как характеристика гравитационно-капиллярных волновых процессов на глубокой воде [2, 157]. Наличие соли-

тонов свойственно информационным системам. Например, передача информации в биологических нейронных сетях осуществляется при помощи уединённых электрохимических волн. В вычислительных системах примером солитона служит некоторая программа, например компьютерный вирус, распространение которой может спровоцировать переход всей вычислительной системы в новое состояние, т.е. вызвать бифуркацию.

Бифуркация проявляется при стохастическом резонансе, когда в нелинейных системах сложение полезного сигнала с шумом приводит к усилению первого. Стохастический резонанс может возникать в нелинейных системах, имеющих несколько стабильных или метастабильных состояний, переход между которыми осуществляется в точках бифуркации. При поступлении в такую систему некоторого полезного сигнала $f(x)$, недостаточного для ее возмущения и преодоления потенциального барьера – точки бифуркации, особую роль может сыграть шум $n(x)$ определённой интенсивности, такой, что его суммарной мощности с полезным сигналом $(f(x)+n(x))$ будет достаточно для преодоления точки бифуркации. При повышении мощности шума переход из одного состояния системы в другое происходит всё более хаотично, так что выходной сигнал становится независимым от полезного сигнала. В работе [8] также показана зависимость поведения системы от спектра шума. Таким образом, в системах, которым свойственен стохастический резонанс, шум может усилить полезный сигнал.

Другой вариант преодоления потенциального барьера в условиях недостаточной исходной энергии возможен в квантовых системах при туннельном эффекте, когда частица обладает одновременно волновыми и корпускулярными свойствами [117].

Модель стохастического резонанса рассмотрена в листинге 3.2. Полезный сигнал на входе системы не превышает 30 % $-0.3;+0.3$ от порогового значения перехода системы в относительно устойчивые состояния -1 и $+1$, следовательно, без дополнительного фактора (которым является шум) система не может перейти из одного состояния в другое. Устойчивость системы (запоминание прежнего состояния) обеспечивается рекурсией для случаев, когда суммарной амплитуды сигнала и шума недостаточно для преодоления порогов. Результаты выполнения программы представлены на рис. 3.6.

Листинг 3.2. Модель стохастического резонанса в нелинейной системе (MATLAB)

```
%=================================================================
clear all;          % удаление переменных
close all;          % закрытие всех окон
clc;                % очистка командного окна
```

```
% определение исходных данных
npoint=1000;      % число точек данных
n=(1:npoint)';    % вектор точек отсчёта
sig=0.3*sin(20*n*pi/npoint); % вектор сигнала
noise=0.5*randn(npoint,1);   % вектор шума
data=sig+noise;              % сумма (сигнал+шум)

result=zeros(npoint,1);      % подготовка вектора результатов
result(1)=10;                % начальное состояние системы

% собственно модель нелинейной системы
% с запоминанием предыдущего состояния (рекурсией)
for i=2:npoint
   if data(i)>1       result(i)=10;
   elseif data(i)<-1  result(i)=-10;
   else               result(i)=result(i-1)-data(i-1)+data(i);
   end
end

% визуализация результата
figure;
subplot(2,2,1); plot(sig(1:end));
title('Полезный сигнал');
subplot(2,2,4); plot(noise(1:end));
title('Случайный шум');
subplot(2,2,2); plot(data(1:end));
title('Сумма (сигнал+шум)');
subplot(2,2,3); plot(result(1:end));
title('Выход системы');
%=================================================================
```

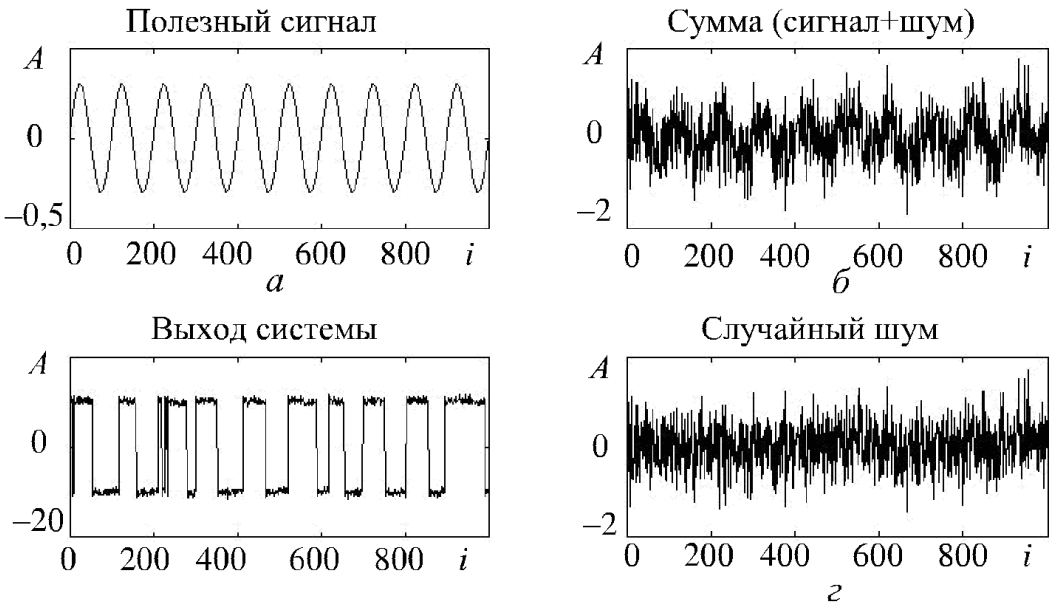


Рис. 3.6. Модель стохастического резонанса

Интересным примером стохастического резонанса, рассмотренным в [173], является сложение видеосигнала со случайным шумом некоторой интенсивности. Нелинейными элементами системы в этом случае будут устройство визуализации (монитор, принтер с нелинейным динамическим диапазоном) и зрительная система наблюдателя.

Стохастический резонанс в бистабильной системе можно описать аналитической моделью вида [61]:

$$\frac{du(x)}{dx} + w(x) = f(x) + n(x)\,, \qquad (3.24)$$

где $u(x)$ – выходной сигнал бистабильной системы; $w(x)$ – нелинейная характеристика системы; $f(x)$ и $n(x)$ – сигнал и шум на входе системы.

Аттрактор Лоренца (или странный аттрактор) также относится к аналитической интерпретации стохастического резонанса в системах со случайными процессами, характерными для бистабильных систем, существующих в реальных физических процессах, например, в слое жидкости, подогреваемой снизу.

3.6. Адаптивные фильтры шума

Достаточно хорошие результаты подавления шума обеспечивают адаптивные фильтры. Из названия понятно, что фильтры такого типа имеют некоторые способности к адаптации, т.е. позволяют производить фильтрацию с учётом динамично изменяющихся особенностей сигнала.

Рассмотрим адаптивный фильтр (рис. 3.7), реализованный по классической схеме [106]. На вход фильтра поступает зашумлённый сигнал $q(x) = f(x) + n(x)$, в суммирующий узел поступают отфильтрованный $u(x)$ и опорный $-d(x)$ сигналы. На основании сигнала ошибки $\Delta(x) = u(x) + (-d(x))$ рассчитываются или корректируются коэффициенты фильтра H. В результате работы системы адаптивной фильтрации получаем сигнал $u(x)$ или сигнал ошибки $\Delta(x)$. В качестве опорного сигнала используют шум или полученный ранее полезный сигнал. Например, адаптивный фильтр, встроенный в систему распознавания голоса, может рассматривать в качестве опорного сигнала шум в паузах между словами. Периодически обновляя опорный сигнал, фильтр перестраивается (адаптируется) в соответствии с изменяющимися характеристиками входного сигнала и шума. Задача адаптивной фильтрации состоит в минимизации значения $\Delta(x)$ [56].

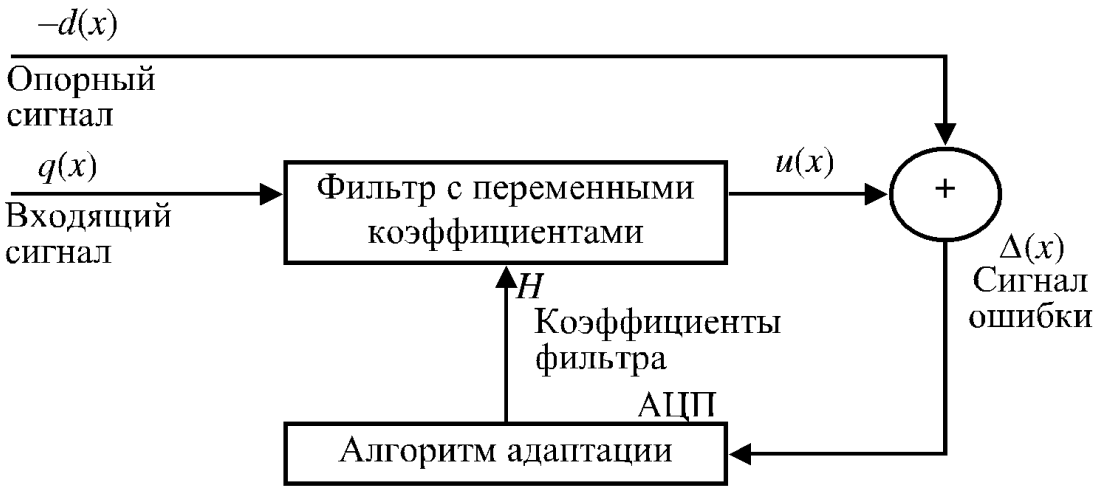


Рис. 3.7. Блок-схема адаптивного фильтра

Адаптивная фильтрация нашла применение в различных научно-практических приложениях, связанных с обработкой и анализом отражённых сигналов, например, в различных видах локации. Различают несколько видов адаптивной фильтрации: адаптивный фильтр Винера, алгоритм наименьших квадратов Уидроу–Хопфа, рекурсивные и статистические схемы.

Адаптивные системы применяют не только для фильтрации, но и для решения других задач, т.к. адаптация служит повышению устойчивости системы.

3.7. О коррекции некоторых искажений оптоэлектронных систем

При сканировании неровно положенный лист или корешок книги могут исказить изображение. Фотоснимки могут содержать искажения, вызванные дефектами преломляющей системой объектива камеры. При фотографировании из космоса близкой к сферической поверхности планеты и отображении её на плоскость ПЗС-матрицы также будут возникать искажения. Подобные искажения могут быть особенно критичны для систем распознавания образов. Любой сигнал, несущий образ некоторого реального объекта, искажён, поскольку создание абсолютной копии возможно лишь на квантовом уровне.

При получении цифровых сигналов с помощью оптических систем возникают специфические искажения – *аберрации*, т.е. отклонения луча в реальной оптической системе от направления луча в идеальной. Качество оптической системы определяется её аберрациями, различают [12]:

– сферическую аберрацию (в частности, наклонного пучка лучей);
– отступление от условия синусов;
– меридиональную и сагиттальную кому (коматическую аберрацию);
– аберрации косых лучей;
– астигматизм;
– кривизну поля;

– дисторсию;
– хроматизм положения;
– хроматизм увеличения;
– хроматическую разность сферических аберраций;
– хроматическую аберрацию наклонного пучка лучей.

В отдельную категорию выделяют искажения, свойственные непосредственно ПЗС-матрицам. Например, *блуминг* – это процесс перетекания заряда с более сильно освещённых пикселов к соседним, менее освещённым. Блуминг проявляется в виде вертикальных полос, расположенных выше и ниже яркого объекта с малыми угловыми размерами.

Устранять искажения желательно на стадии формирования сигнала за счет повышения качества оптоэлектронной системы. Если оптоэлектронная система не может обеспечить приемлемого качества цифрового сигнала, непосредственно к цифровым сигналам применяют специальные численные методы коррекции. Одним из распространенных видов искажений является оптическая дисторсия, проявляющаяся в том, что при прохождении света через реальную оптическую систему положение точек, образующих изображение, смещается относительно того положения, которое должно быть в идеальной оптической системе. Радиальная дисторсия обусловлена сферической поверхностью линз объектива, а тангенциальная – неперпендикулярностью главной оптической оси к плоскости изображения и прохождением главной оптической оси не через центр кадра [33].

Связь координат точек, полученных в реальной (x, y) и идеальной (x_0, y_0) системах, может быть выражена следующими уравнениями:

$$\begin{cases} x_0 = x + \delta_x, \\ y_0 = y + \delta_y. \end{cases} \tag{3.25}$$

При этом суммарное влияние радиальной и тангенциальной дисторсии определяется как

$$\begin{cases} \delta_x = x\ k_1(r^2 - r_0^2) + k_2(r^4 - r_0^4)\ + (r^2 + 2x^2)p_1 + 2xyp_2, \\ \delta_y = y\ k_1(r^2 - r_0^2) + k_2(r^4 - r_0^4)\ + 2xyp_1 + (r^2 + 2y^2)p_2, \end{cases} \tag{3.26}$$

где k_1, k_2 и p_1, p_2 – коэффициенты радиальной и тангенциальной дисторсии; $r = \sqrt{x^2 + y^2}$ – расстояние до точки; r_0 – расстояние до точки нулевой дисторсии.

Отметим, что в общем случае к *дисторсии* относят разнообразные искажения, связанные с изменением формы сигнала. Например, реактивные

свойства электронных устройств, обусловленные наличием емкостей и индуктивностей, способствуют «сглаживанию» аналогового сигнала, т.е. фактически производят подавление высокочастотной составляющей.

В цифровых сигналах, полученных с помощью оптических систем, скорректировать дисторсию возможно с помощью уравнений, подобных (3.26), коэффициенты которых определяют опытным путём при помощи специальных тестовых изображений [27, 88, 151, 176]. В работах представлены [26, 76] интересные предложения по решению задач калибровки оптических систем и устранения искажений при помощи нейронных сетей.

Следующий вид искажения проявляется в недостаточной резкости изображения, вызванной *сферической* и *коматической* аберрациями оптической системы. Недостаточная резкость изображения также может быть обусловлена свойствами среды, в которой распространяется свет. При сферической и коматической аберрациях оптической системы для повышения резкости изображения перед применением корректирующих уравнений необходимо оценить параметры конкретной оптической системы. При сферической аберрации проходящие через оптическую систему лучи от точечного источника вместо точки образуют некоторую размытую область. Моделью сферической аберрации является свёртка, при этом ФРТ определяется кривизной поверхности и оптической силой линз, образующих оптическую систему. Соответственно корректировать искажения, вызванные сферической аберрацией, можно с помощью *обратной свёртки* при известной ФРТ. Радиус функции рассеяния точки при коррекции сферической аберрации [15] возможно определить, например, согласно:

$$r_c = \frac{r_m(k_1 r_m^{\,2} + k_2 r_m^{\,4})}{f - (k_1 r_m^{\,2} + k_2 r_m^{\,4})}, \qquad (3.27)$$

где k_1, k_2 – коэффициенты сферической аберрации, постоянные для каждой оптической системы; f – фокусное расстояние; r_m – радиус диафрагмы.

В случае коматической аберрации, в отличие от сферической, ФРТ несимметрична по направлениям, при этом радиус функции рассеяния точки можно оценить [116], например, при помощи выражения:

$$r_k = \frac{1}{3} Br(3r_x^2 + r_y^2), \qquad (3.28)$$

где B – коэффициент комы, постоянный для каждой оптической системы; r – расстояние от точки до главной оптической оси M, r_x и r_y – зрачковые координаты крайнего луча.

В случае сферической аберрации ФРТ может быть заменена двумерной функцией распределения Гаусса. Пакет MATLAB предлагает специаль-

ные инструменты для выполнения операции обратной свёртки (листинг 3.3) [134, 150].

Листинг 3.3. Моделирование: размытие и коррекция (MATLAB)

```
%=================================================================
clear all;        % удаление переменных
close all;        % закрытие всех окон
clc;              % очистка командного окна

% формирование исходного изображения
Image=checkerboard(8);
% функция рассеяния точки
PSF=fspecial('gaussian',8,7);

% искажение изображения: размытие и шум
me=0;       % медиана шума
de=0.0001; % дисперсия шума
BlurredNoisy=imnoise(imfilter(Image,PSF),'gaussian',me,de);
BlurredNoisy=imfilter(Image,PSF);

% восстановление изображения обратной свёрткой
ni=15; % число итераций
ReIm=deconvlucy(BlurredNoisy,PSF,ni,sqrt(de));

% визуализация результатов
figure; colormap(Gray);
subplot(2,2,1); imshow(Image);
title('Исходное изображение');
subplot(2,2,2); imshow(BlurredNoisy);
title('Искажённое изображение');
subplot(2,2,3); imagesc(PSF);
title('Функция рассеяния точки');
subplot(2,2,4); imshow(ReIm);
title('Восстановленное изображение');
%=================================================================
```

На рис. 3.8 приведены результаты выполнения программы (см. листинг 3.3). Очевидно, что при повышении резкости изображения полного восстановления не происходит, это связано с потерей части информации при свёртке.

Соответственно при размытии, обусловленном оптической системой, условиями приёма или обработки сигнала, с помощью обратной свёртки также невозможно получить качество «неискажённого» сигнала, но в некоторых случаях возможно повысить качество сигнала до уровня, достаточного для дальнейшей обработки и анализа.

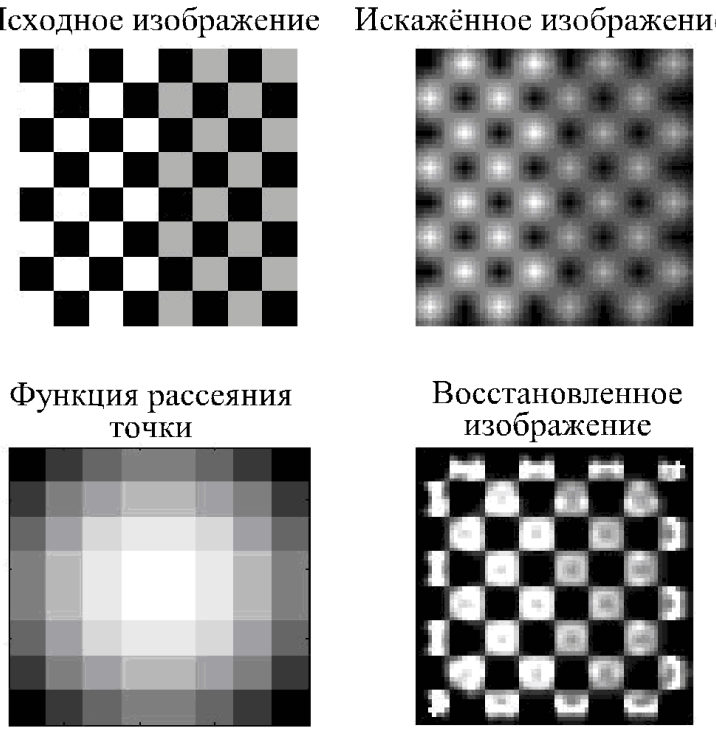


Рис. 3.8. Устранение сферической аберрации

Хроматическая аберрация возникает за счет разложения в оптической системе света на спектр (дисперсии), видимым результатом такого искажения является появление ореолов, особенно вокруг контрастных границ объектов и, как следствие, уменьшение контрастности изображения. Обычно уровень этого искажения снижается до допустимого повышением качества оптической системы. Способы электронной коррекции подобных искажений основаны на том, что в (цветных) оптоэлектронных системах изображения регистрируются по трём цветовым каналам RGB, т.е. в области ореола изображение в каждом цветовом канале будет смещено на некоторую величину.

Астигматизм – явление, при котором лучи одного и того же пучка, идущие по отношению друг к другу в двух взаимно перпендикулярных плоскостях, после преломления в оптической системе не собираются в одну точку, а образуют две точки (иногда более).

Кривизна поля изображения – аберрация, в результате которой изображение плоского объекта, перпендикулярного оптической оси объектива, лежит на поверхности, вогнутой либо выпуклой к объективу.

В общем случае можно рассматривать следующую линейную модель оптического приёмника (рис. 3.9). На вход оптического приёмника поступает некоторый сигнал $f(x, y)$, первым входным модулем модели является оптическая система (ОС) образованная совокупностью оптических преобразователей (системой линз, зеркал, призм, фильтров и т.д.); следующим элементом является детектор оптического сигнала (ДОС), например,

ПЗС-матрица; затем аналоговый сигнал преобразуется, например, усиливается; далее сигнал преобразуется в цифровой вид при помощи модуля АЦП. Элементы системы (ОП – оптический преобразователь; АП – аналоговый преобразователь, ДОС) обладают собственной импульсной характеристикой ФРТ (PSF). В результате на выходе приёмника будет дискретный сигнал $f[x, y]$, являющийся цифровым образом исходного сигнала, свёрнутого с импульсными характеристиками модулей модели:

$$u(x, y) = \mathrm{PSF}_{\mathrm{ОП}} * \mathrm{PSF}_{\mathrm{ДОС}} * \mathrm{PSF}_{\mathrm{АП}} * f(x, y),$$
$$f[x, y] = \mathrm{АЦП}\ \ u(x, y)\ . \qquad (3.29)$$

Рис. 3.9. Модель оптического приёмника

Учитывая специфику импульсной характеристики каждого модуля, возможно производить коррекцию сигнала. Подобный приём рассмотрен в [125]. На практике измерение полной ФРТ является достаточно сложной задачей, т.к. необходимо учитывать влияние пространственной дискретизации, эффекты, связанные со смещением изображения в процессе интегрирования сигнала детектором. Также существует вероятность изменения ФРТ с течением времени и в результате физических воздействий на оптический приёмник, например, изменение влажности, температуры, давления, химического состава окружающей среды и пр.

Любое приёмное устройство, кроме пространственных, вносит искажения в энергетические характеристики сигнала. Например, цвета изображения, полученного при помощи фотоаппарата, не будут в точности соответствовать истинным цветам объекта фотографирования. Причина этих искажений состоит в том, что приёмник регистрирует отклик на исходный сигнал, в свою очередь, отклик синтезируется датчиком в результате воздействия на него исходного сигнала. Сам датчик при этом обладает определёнными физическими характеристиками, например, спектром поглощения. В результате принятый сигнал будет содержать информацию не только об исходном сигнале, но и обо всём приёмном тракте. Искажения сигнала возможно рассматривать с точки зрения преобразования энергии.

Для цифрового графического изображения цвет – это лишь набор некоторых цифровых кодов, но, преобразовав эти коды в электронные импульсы и подав на устройство визуализации (например, монитор), можно получить цветную картинку, которая будет дополнительно искажена влиянием особенностей конкретного устройства визуализации. При регистрации цифровых сигналов принято говорить не об абсолютных энергетических характеристиках, а об относительных и условных, принятых в системе

данного приёмно-измерительного устройства и зарегистрированных в результате отклика датчиков устройства на внешний сигнал.

В специализированной литературе [33, 63] можно найти разнообразные способы коррекции различных искажений, в зависимости от типа приёмного устройства и условий приёма сигнала. Практическая реализация значительной части таких корректирующих способов ориентирована на конкретные устройства, поэтому их рассмотрение выходит за рамки данного труда.

4. Декомпозиция, синтез цифровых сигналов

Декомпозиция и синтез – наиболее распространённые операции цифровой обработки сигналов. Многие методы анализа данных и распознавания образов построены на способах декомпозиции и синтеза. Современные методы компрессии, такие как *JPEG* (разработчик *Joint Photographic Experts Group*) [84, 105, 112], *MPEG* (*Moving Picture Experts Group*) [108, 153, 154], используют декомпозицию и синтез на базе дискретного косинусного или вейвлет-преобразования.

4.1. Линейные системы

Системы называются *линейными*, если они обладают свойствами *однородности* и *аддитивности*.

Аддитивность системы F означает, что при воздействии на её вход суммы сигналов $f_1[R^n]+f_2[R^n]$ на выходе возникает сигнал $u_1[R^n]+u_2[R^n]$, причём $u_1[R^n]$ есть результат воздействия сигнала $f_1[R^n]$, а $u_2[R^n]$ – сигнала $f_2[R^n]$:

$$F:\left[f_1[R^n]+f_2[R^n]\right]\rightarrow\left[u_1[R^n]+u_2[R^n]\right]. \qquad (4.1)$$

Однородность системы F означает, что входной сигнал $f[R^n]$ вызовет ответную реакцию системы, т.е. выходной сигнал:

$$u[R^n]=hf[R^n], \qquad (4.2)$$

где h =const в общем случае может быть *собственной* многомерной *импульсной характеристикой* системы $h[R^n]$, тогда выражение (4.2) будет иметь вид:

$$f[R^n]*h[R^n]=u[R^n], \qquad (4.3)$$

операция $(*)$ называется *свёрткой*, её мы рассмотрим далее.

Таким образом, линейная система подразумевает преобразования сигналов путём их сложения и/или умножения на константу (массив констант), в общем случае – свёртку с собственной импульсной характеристикой системы, в противном случае система не является линейной. Любую линейную систему можно представить в виде совокупности подсистем, производящих умножение сигналов на константу и сложение. Линейным системам свойственна коммутативность, т.е. если для преобразования сигнала используется цепочка последовательно соединённых линейных систем, то порядок соединения не влияет на результат.

Превалирующее число задач ЦОС можно решить выполняется с помощью операций, осуществляемых линейными системами. Нелинейные системы в большинстве случаев можно представить в виде совокупности

линейных систем. В отличие от линейных систем, на выходе нелинейных возникают частотные компоненты, не входящие в частотные спектры исходных сигналов [78].

4.2. Понятие свёртки

Свёртка является математической бинарной операцией, формирующей из двух входных сигналов один, один сигнал на входе операции называют *импульсной характеристикой* системы, другой – входным сигналом (рис. 4.1). Свёртку можно описать выражением:

$$f[R^n] * h[R^n] = u[R^n], \tag{4.4}$$

где $f[R^n]$ – входной сигнал; $h[R^n]$ – импульсная характеристика системы; $u[R^n]$ – выходной сигнал.

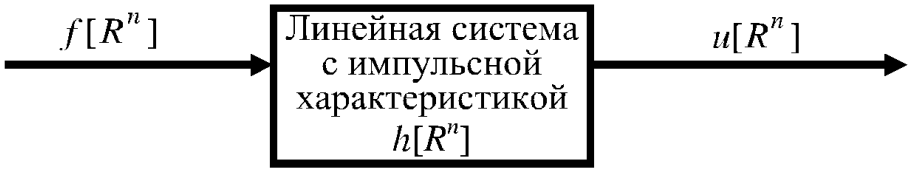


Рис. 4.1. Блок-схема свёртки

Согласно определению, *импульсной характеристикой* $h[R^n]$ называется реакция линейной дискретной системы на цифровой единичный импульс $f_0[R^n]$, поданный на вход при нулевых начальных условиях.

Переходной характеристикой $g[R^n]$ называется реакция линейной дискретной системы на сигнал перехода (между двумя соседними отсчетами) из нулевых уровней в единичные $f_{01}[R^n]$ или из единичных в нулевые $f_{10}[R^n]$, поданный на вход этой системы при нулевых начальных условиях:

$$g(R^n) = \int_{R^n} h[R^n]. \tag{4.5}$$

Свёртка может быть применена для самых разнообразных преобразований цифровых сигналов, с ее помощью можно произвести декомпозицию и синтез. Свёртку можно рассматривать как один из видов корреляционного анализа, например, она может помочь в распознавании заранее известных образов изображения, которые используются в качестве импульсных характеристик системы.

В дискретном виде свёртку одномерного сигнала можно описать уравнением:

$$\begin{cases} u[i] = \sum_{j=0}^{L-1} h[j] f[i-j], \\ 0 \le i - j \le N - 1, \end{cases} \tag{4.6}$$

где $h[]$ – импульсная характеристика системы размером L, $f[]$ – входной сигнал размером N; $u[]$ – выходной сигнал (результат свёртки) размером $N + L - 1$.

В двумерном случае уравнение свёртки (4.6) будет иметь вид:

$$\begin{cases} u[i_0, i_1] = \sum_{j_0=0}^{L_0-1} \sum_{j_1=0}^{L_1-1} h[j_0, j_1] f[i_0 - j_0, i_1 - j_1], \\ 0 \le i_0 - j_0 \le N_0 - 1, \\ 0 \le i_1 - j_1 \le N_1 - 1, \end{cases} \tag{4.7}$$

где $h[]$ – импульсная характеристика системы $L_0 \times L_1$, $f[]$ – входной сигнал $N_0 \times N_1$; $u[]$ – выходной сигнал размером $(N_0 + L_0 - 1) \times (N_1 + L_1 - 1)$.

В случае n-мерного сигнала уравнение свёртки будет иметь вид:

$$\begin{cases} u[i_0, i_1, ..., i_{n-1}] = \sum_{j_0=0}^{L_0-1} \sum_{j_1=0}^{L_1-1} ... \sum_{j_{n-1}=0}^{L_{n-1}-1} h[j_0, j_1, ..., j_{n-1}] f[i_0 - j_0, i_1 - j_1, ..., i_{n-1} - j_{n-1}], \\ 0 \le i_0 - j_0 \le N_0 - 1, \\ 0 \le i_1 - j_1 \le N_1 - 1, \\ ... \\ 0 \le i_{n-1} - j_{n-1} \le N_{n-1} - 1. \end{cases} \tag{4.8}$$

Свёртка просто реализуется программно, при этом ее вычислительная сложность может быть достаточно большой.

Вычислительную сложность алгоритма свёртки можно оценить как:

$$O(\prod_{i=0}^{n-1} N_i \prod_{i=0}^{n-1} L_i) = O(\prod_{i=0}^{n-1} N_i L_i), \; i = 0, ..., n-1. \tag{4.9}$$

Из выражения (4.9) видно, что сложность вычислений зависит от размерности пространства, размеров входного сигнала и импульсной характеристики. Уменьшение вычислительной сложности возможно благодаря снижению размеров импульсной характеристики системы. При постоянстве размеров импульсной характеристики в соответствии с асимптотической нотацией выражение (4.9) можно сократить:

$$O(\prod_{i=0}^{n-1} N_i)\,. \qquad (4.10)$$

Свёртка удовлетворяет следующим законам:

– переместительный:

$$f[R^n] * h[R^n] = h[R^n] * f[R^n]\,; \qquad (4.11)$$

– сочетательный:

$$f[R^n] * h[R^n] * x[R^n] = f[R^n] * \left[h[R^n] * x[R^n] \right]; \qquad (4.12)$$

– распределительный:

$$f[R^n] + h[R^n] \; * x[R^n] = f[R^n] * x[R^n] + h[R^n] * x[R^n]\,. \qquad (4.13)$$

4.3. Программная реализация свёртки

Рассмотрим программную реализацию свёртки в одномерном случае (листинг 4.1). Массив `data[]` содержит исходный одномерный сигнал, размерность массива `data[]- sd`. Массив `pulse[]` содержит импульс, которым производится свёртка, размер массива `pulse[]- sp`. Результат помещается в массив `result[]` размером `sd+sp-1`.

Листинг 4.1. Одномерная свёртка сигнала (C++)

```
//=============================================================
...
// цикл по сигналу
for(int i=0; i<sd; i++) {
  // цикл по импульсу
  for(int j=0; j<sp; j++) {
    result[i+j]+=data[i]*pulse[j];
  }
}
...
//=============================================================
```

Рассмотрим свёртку двумерного сигнала – изображения $N_0 \times N_1$ с импульсной характеристикой $L_0 \times L_1$. Отметим, что двумерные импульсные характеристики системы в дискретном виде принято называть *функцией рассеяния точек* (ФРТ).

Фрагмент программы, производящей свёртку двумерного сигнала, приведен в листинге 4.2. Массив `data2d[][]` содержит исходный двумерный сигнал, размерность массива `data2d[][]- sdY` и `sdX` по строкам и столбцам соответственно. Массив `pulse2d[][]` содержит ФРТ, с которой производится свёртка, размер массива `pulse2d[][]- spY` и `spX` по строкам и столбцам соответственно. Результат помещается в массив `result2d[][]` размером `sdY+spY-1` и `sdX+spX-1`.

Листинг 4.2. Двумерная свёртка сигнала (C++)

```
//=============================================================
...
int ix, iy;
// выполнение свёртки
// циклы по сигналу
for(int iyd=0; iyd<sdY; iyd++) {
  for(int ixd=0; ixd<sdX; ixd++) {
    // циклы по импульсу ФРТ
    for (int iyp=0; iyp<spY; iyp++) {
      iy=iyd+iyp;
      for(int ixp=0; ixp<spX; ixp++) {
        ix=ixd+ixp;
        result2d[iy][ix]+=pulse2d[iyp][ixp]*data2d[iyd][ixd];
      }
    }
  }
}
...
//=============================================================
```

На практике цифровые сигналы зачастую являются многоканальными, например, изображения в формате RGB содержат три цифровых канала, поэтому свёртку в большинстве случаев выполняют для каждого из них. По каждому каналу может быть использована своя ФРТ, связанная, например, с особенностями восприятия человеком различных цветов в силу специфики анатомического строения глаза или с особенностями самого изображения.

Очевидно, что из-за специфики цифровых сигналов – ограниченности разрядной сетки и дискретности значений при свёртке, как и при многих других преобразованиях, – возможны потеря значимой информации и выход за допустимый диапазон значений. Например, для цветовой палитры RGB, по каждому каналу, диапазон значений 0–255, дискретный шаг равен единице. Эти особенности необходимо учитывать при разработке и реализации вычислительных алгоритмов, производить нормировку динамического диапазона.

Быстродействие алгоритма может быть обеспечено вычислениями в целочисленных форматах и совмещением в едином циклическом пакете свёртки по всем каналам цифрового сигнала. Как было сказано, вычислительная сложность алгоритма зависит от размеров исходного сигнала и ФРТ. Для обработки изображений в большинстве случаев достаточно ФРТ размером 3×3 или 5×5.

Размеры полученного в результате свёртки сигнала больше размеров исходного сигнала и импульсной характеристики, при этом в большинстве случаев интерес представляет центральная часть, которая может

быть получена путём вырезания из результирующего сигнала части, совпадающей по размерам с исходным сигналом.

Свёртка является одним из самых распространённых способов обработки цифровых сигналов, она используется для обработки статических и динамических изображений. Вычисление корреляции на базе свёртки может быть полезно, например, при распознавании образов. Например, при фрактальном сжатии изображений и вычислении автокорреляции на базе свёртки возможно найти самоподобные элементы.

4.4. Примеры обработки многомерных сигналов с помощью свёртки

В цифровой обработке сигналов свёртка используется очень широко – самостоятельно и как составная часть некоторых методов. Свёртка является своеобразным математическим обобщением, с её помощью можно решать множество задач преобразования сигналов. На базе свёртки, выбирая различные импульсные характеристики, можно реализовать значительное количество прочих операций, рассмотрим некоторые из них.

Для двумерного непрерывного (аналогового) сигнала $f(x, y)$ производные в направлениях x и y вычисляются соответственно как $\frac{\partial f(x, y)}{\partial x}$ и $\frac{\partial f(x, y)}{\partial y}$. Переходя к дискретному образу $f[x, y]$ непрерывной функции $f(x, y)$, полагаем, что ее приращения имеют значения: $\partial_x f(x, y) \approx f[x+1, y] - f[x, y]$ и $\partial_y f(x, y) \approx f[x, y+1] - f[x, y]$; приращение аргументов $\partial x = \partial y \approx \Delta x = \Delta y = 1$, тогда производные сигнала $f[x, y]$:

– в направлении x

$$\frac{\partial f(x, y)}{\partial x} \approx f[x+1, y] - f[x, y],$$

– в направлении y

$$\frac{\partial f(x, y)}{\partial y} \approx f[x, y+1] - f[x, y].$$

Производные сигнала $f[x, y]$ можно вычислить с помощью свертки с ФРТ:

– в направлении x

$$h[x, y] = \begin{bmatrix} 0 & 0 & 0 \\ 0 & 1 & -1 \\ 0 & 0 & 0 \end{bmatrix} \text{ или } h[x, y] = \begin{bmatrix} 0 & 0 & 0 \\ 1 & -1 & 0 \\ 0 & 0 & 0 \end{bmatrix}, \qquad (4.14)$$

– в направлении y

$$h[x,y]=\begin{bmatrix}0&0&0\\0&1&0\\0&-1&0\end{bmatrix} \text{ или } h[x,y]=\begin{bmatrix}0&1&0\\0&-1&0\\0&0&0\end{bmatrix}. \qquad (4.15)$$

Вычислить сумму производных по направлениям $\frac{\partial f(x,y)}{\partial x}+\frac{\partial f(x,y)}{\partial y}$ можно с помощью свёртки с ФРТ:

$$h[x,y]=\begin{bmatrix}0&1&0\\1&-2&0\\0&0&0\end{bmatrix} \text{ или } h[x,y]=\begin{bmatrix}0&0&0\\0&2&-1\\0&-1&0\end{bmatrix}. \qquad (4.16)$$

Вычислим производные второго порядка $\frac{\partial^2 f(x,y)}{\partial x^2}$ и $\frac{\partial^2 f(x,y)}{\partial y^2}$ для $f[x,y]$ с помощью свёртки с ФРТ:

$$h[x,y]=\begin{bmatrix}0&0&0\\1&-2&1\\0&0&0\end{bmatrix} \text{ или } h[x,y]=\begin{bmatrix}0&1&0\\0&-2&0\\0&1&0\end{bmatrix}. \qquad (4.17)$$

Для функции $f[x,y]$ получим суммы производных второго порядка:

$$h[x,y]=\begin{bmatrix}0&1&0\\1&-4&1\\0&1&0\end{bmatrix}. \qquad (4.18)$$

С помощью свёртки можно выполнить вычисления производных и более высоких порядков, например, решить уравнения в частных производных на массиве граничных условий.

Свёртка позволяет реализовать разнообразные фильтры, причём для высокочастотных достаточно использовать ФРТ небольших размеров.

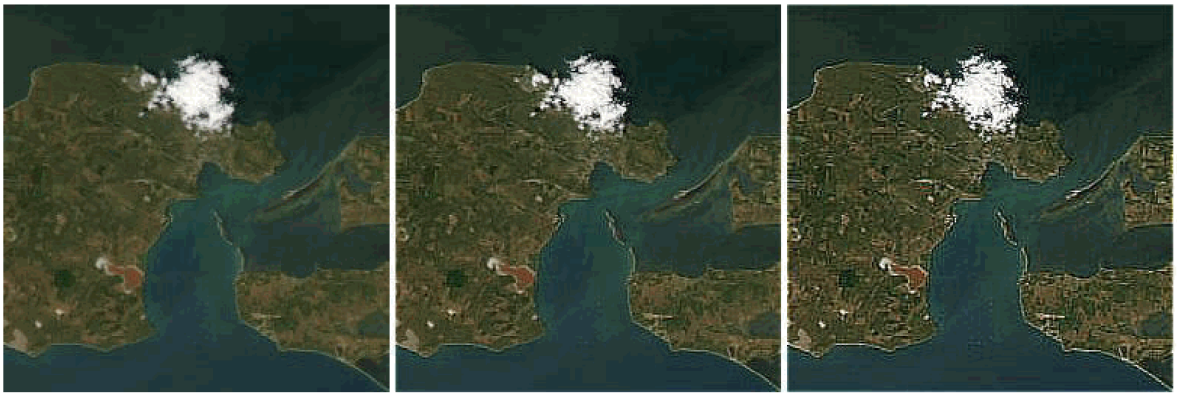
Высокочастотный фильтр подчёркивания границ на базе ФРТ 3×3 :

$$h[x,y]=\begin{bmatrix}-0.125&-0.125&-0.125\\-0.125&2&-0.125\\-0.125&-0.125&-0.125\end{bmatrix}, \qquad (4.19)$$

а на базе ФРТ 5×5 реализует фильтр повышения контрастности изображения:

$$h[x, y] = \begin{bmatrix} -0.0125 & -0.0125 & -0.0125 & -0.0125 & -0.0125 \\ -0.0125 & -0.1 & -0.1 & -0.1 & -0.0125 \\ -0.0125 & -0.1 & 2 & -0.1 & -0.0125 \\ -0.0125 & -0.1 & -0.1 & -0.1 & -0.0125 \\ -0.0125 & -0.0125 & -0.0125 & -0.0125 & -0.0125 \end{bmatrix}. \quad (4.20)$$

На рис. 4.2 приведён пример использования свёртки с ФРТ (4.20) для повышения контрастности фрагмента снимка поверхности Земли, полученного со спутника AQUA.



а *б* *в*

Рис. 4.2. Повышение контрастности спутникового снимка Земли с помощью свёртки: *а* – исходное изображение; *б* – свёртка выполнена один раз, *в* – дважды

Свёртка сигналов находит применение в самом широком спектре научно-технических и прикладных задач. На базе концептуальной сущности свертки можно реализовать различные математические методы обработки и анализа n-мерных сигналов.

4.5. Декомпозиция и синтез в линейных системах

Декомпозиция в линейных системах – представление одного, сколь угодно сложного, сигнала в виде суммы нескольких, относительно простых:

$$D: f[R^n] \rightarrow u_0[R^n], u_1[R^n], ..., u_{p-1}[R^n], \quad (4.21)$$

где D – операция декомпозиции; $f[R^n]$ – исходный сигнал; $u_0[R^n], u_1[R^n], ..., u_{p-1}[R^n]$ – множество элементов (от 0 до $p-1$) декомпозиции исходного сигнала в n-мерном пространстве.

Обратный декомпозиции процесс D^{-1} восстановления сигнала путём суммирования элементов называется *синтезом*:

$$f[R^n] = \sum_{i=0}^{p-1} D^{-1} \; u_i[R^n]. \quad (4.22)$$

Декомпозицию и синтез сигнала можно выполнять в пространстве и/или времени. Цель преобразований и особенности сигнала определяют способ декомпозиции и синтеза.

4.6. Фурье-декомпозиция и синтез

Одной из классических операций декомпозиции и синтеза одномерного сигнала является *преобразование Фурье*[6].

Взаимно-ортогональные функции синуса и косинуса, выступающие в качестве компонентов фурье-декомпозиции, называют *базисными функциями* преобразования Фурье. Различают преобразование Фурье непрерывных и дискретных функций [31].

Для непрерывной функции $f(x)$, заданной в замкнутом промежутке $(\theta, \theta + 2\pi)$ и не имеющей в нём либо имеющей ограниченное число экстремумов, ряд Фурье сходится всюду и его сумма равна $f(x)$ для всякого значения $x \in (\theta, \theta + 2\pi)$, на концах промежутка $(\theta, \theta + 2\pi)$ сумма ряда имеет значение $\frac{1}{2}(f(\theta) + f(2\pi))$. Косинусные и синусные коэффициенты ряда Фурье могут быть вычислены в соответствии с выражениями [5]:

$$\begin{cases} A_n = \dfrac{1}{\pi} \displaystyle\int_{\theta}^{\theta+2\pi} f(x)\sin(nx)dx, \\ B_n = \dfrac{1}{\pi} \displaystyle\int_{\theta}^{\theta+2\pi} f(x)\cos(nx)dx, \\ A_0 = 0, \\ B_0 = \dfrac{1}{2\pi} \displaystyle\int_{\theta}^{\theta+2\pi} f(x)dx, \end{cases} \tag{4.23}$$

где $f(x)$ – функция разложения, A_n, B_n – коэффициенты ряда Фурье с индексом $n = 1, 2, \ldots$

Обратное преобразование Фурье (синтез) может быть выполнено с помощью выражения:

$$f(x) = B_0 + \sum_{n=1}^{\infty} A_n \sin(nx) + B_n \cos(nx) \quad . \tag{4.24}$$

---

[6] Жан Батист Жозеф Фурье (21 марта 1768, Осер – 16 мая 1830, Париж) – математик и физик.

Если период равен не 2π, а произвольному T, то функцию $f(x)$ следует рассматривать на интервале $(\theta,\theta+T)$, при этом выражения (4.23) и (4.24) приобретут вид:

$$\begin{cases} A_n = \dfrac{2}{T}\int\limits_{\theta}^{\theta+T} f(x)\sin(n\omega x)dx, \\ B_n = \dfrac{2}{T}\int\limits_{\theta}^{\theta+T} f(x)\cos(n\omega x)dx, \\ A_0 = 0, \\ B_0 = \dfrac{1}{T}\int\limits_{\theta}^{\theta+T} f(x)dx, \end{cases} \tag{4.25}$$

$$f(x) = B_0 + \sum_{n=1}^{\infty} A_n \sin(n\omega x) + B_n \cos(n\omega x) \quad , \tag{4.26}$$

где $\omega = \dfrac{2\pi}{T}$.

Используя формулы Эйлера[7] [22]:

$$\begin{cases} \cos(nx) = \dfrac{e^{jnx} + e^{-jnx}}{2}, \\ \sin(nx) = \dfrac{e^{jnx} - e^{-jnx}}{2j}, \end{cases} \tag{4.27}$$

где $j = \sqrt{-1}$, можно записать преобразование Фурье на множестве комплексных чисел.

Для обратного преобразования Фурье получим:

$$f(x) = \sum_{n=1}^{\infty} \frac{A_n}{2j}(e^{jn\omega x} - e^{-jn\omega x}) + \sum_{n=1}^{\infty} \frac{B_n}{2}(e^{jn\omega x} + e^{-jn\omega x}) + B_0 =$$

$$= \sum_{n=1}^{\infty}\left(\frac{B_n - jA_n}{2} e^{in\omega x} + \frac{B_n + jA_n}{2} e^{-in\omega x} \right) + B_0,$$

далее с учётом (4.25) и (4.26) можно записать:

$$\frac{B_n - jA_n}{2} = \frac{1}{T}\int\limits_{\theta}^{\theta+T} f(x)\ \cos(n\omega x) - j\sin(n\omega x)\ dx = \frac{1}{T}\int\limits_{\theta}^{\theta+T} f(x)e^{-jn\omega x}dx,$$

[7] Леонард Эйлер (4 апреля 1707, Базель, Швейцария – 7 сентября 1783, Санкт-Петербург) – математик, внёсший значительный вклад в развитие математики, механики, физики, астрономии и ряда прикладных наук.

$$\frac{B_n + jA_n}{2} = \frac{1}{T}\int_{\theta}^{\theta+T} f(x)e^{jn\omega x}dx\,.$$

Согласно работе [5]:

– из выражения для $\frac{B_n - jA_n}{2}$ можно получить выражение для $\frac{B_n + jA_n}{2}$, заменив n на $-n$, при этом если первый коэффициент обозначить через C_n, то второй должен быть обозначен как C_{-n};

– постоянный член можно записать в виде: $B_0 = \frac{1}{T}\int_{\theta}^{\theta+T} f(x)e^0 dx$.

Следовательно, для преобразования Фурье на множестве комплексных чисел можно записать:

$$f(x) = \sum_{n=-\infty}^{n=+\infty} C_n e^{jn\omega x}\,, \tag{4.28}$$

где комплексные коэффициенты Фурье:

$$C_n = \frac{1}{T}\int_{\theta}^{\theta+T} f(x)e^{-jn\omega x}dx\,. \tag{4.29}$$

Для дискретных функций, заданных в виде отсчётов, выражения (4.25) и (4.26) можно записать в виде:

$$\begin{cases} A[k] = \frac{2}{N}\sum_{n=0}^{N-1} f[n]\sin[\frac{2\pi kn}{N}], \\ B[k] = \frac{2}{N}\sum_{n=0}^{N-1} f[n]\sin[\frac{2\pi kn}{N}], \\ A[0] = 0, \\ B[0] = \frac{1}{N}\sum_{n=0}^{N-1} f[n], \end{cases} \tag{4.30}$$

$$f[n] = B[0] + \sum_{k=1}^{K-1}\left(A[k]\sin[\frac{2\pi kn}{N}] + B[k]\cos[\frac{2\pi kn}{N}]\right) \tag{4.31}$$

или на множестве комплексных чисел:

$$f[n] = \sum_{k=0}^{K-1} C[k]e^{j\frac{2\pi nk}{N}}\,, \tag{4.32}$$

где

$$C[k] = \frac{1}{N}\sum_{n=0}^{N-1} f[n]e^{-j\frac{2\pi nk}{N}}\,. \tag{4.33}$$

Схожесть выражений для обратного (4.32) и прямого (4.33) преобразования именуют дуальностью преобразования Фурье.

Описанный способ получения частотных коэффициентов может быть применён в случае разложения функции в более общие ряды декомпозиции по системе ортогональных базисных функций.

Отметим, что в соответствии с теоремой Парсеваля[8] энергия функции $f(x)$, представленной во времени или пространстве, равна интегральной составляющей энергии частотного спектра $F\ \ f(x)$:

$$\int_{-\infty}^{+\infty} |f(x)|^2\, dx = \int_{-\infty}^{+\infty} |F\ \ f(x)\ |^2\, df\ , \tag{4.34}$$

где $F\ \ f(x)$ – непрерывное преобразование Фурье сигнала $f(x)$.

Для дискретного преобразования Фурье $F[k]$ сигнала $f[n]$ выражение (4.17) будет иметь вид:

$$\sum_{n=0}^{N-1} f^2[n] = \frac{1}{N} \sum_{k=0}^{N-1} F^2[k]\,. \tag{4.35}$$

В практических целях используют *быстрое преобразование Фурье* (БПФ). Идея БПФ была сформулирована Гауссом[9] много раньше, чем появилась возможность практической реализации с помощью технических средств. Вторую жизнь в идею вдохнули Дж. Кули и Дж. Тьюки в 1965 г. [139]. Вычислительная сложность алгоритма БПФ оценивается:

$$O(n\log(n))\,, \tag{4.36}$$

где n – размер массива для преобразования Фурье.

В современных системах ЦОС БПФ – один из наиболее используемых методов. Существует множество модификаций БПФ, некоторые из них ориентированы на специальные типы данных, другие универсальные. Способы практического применения и особенности различных видов преобразования Фурье подробно рассмотрены, например, в работах [85, 95, 112, 114] и др.

4.7. Оконное преобразование Фурье

Оконное преобразование Фурье выполняется выбором некоторого периода $T = (t_{\min}, t_{\max})$ для функции $f(x)$, заданной в интервале $X = (x_{\min}, x_{\max})$, причём $T \subset X$:

---

[8] Парсеваль Марк-Антуан (27 апреля 1755, Розьер-о-Салин – 16 августа 1836, Париж) – математик.

[9] Гаусс Иоганн Карл Фридрих (30 апреля 1777, Брауншвейг – 23 февраля 1855, Гёттинген) – математик, астроном и физик.

$$C_n(\tau) = \frac{1}{T}\int_{\theta}^{\theta+T} f(x)w(x-\tau)e^{-jn\omega x}dx\,, \qquad (4.37)$$

где $w(x-\tau)$ – некоторая оконная функция, τ – переменная, определяющая сдвиг окна, прочие обозначения те же, что и в предыдущем разделе. С помощью *оконного преобразования Фурье* можно получить образ функции $f(x)$ в *фазовом пространстве*. Следует отметить, что в качестве оконных обычно используют функции, имеющие всплески: некоторый однополярный ограниченный и быстро затухающий по мере удаления от области наибольшей амплитуды.

Рассмотрим примеры таких функций:

– прямоугольное окно (окно Дирихле):

$$w(x) = \begin{cases} 1, x \in [x_{\min}, x_{\max}] \\ 0, x \notin [x_{\min}, x_{\max}] \end{cases}, \qquad (4.38)$$

где $[x_{\min}, x_{\max}]$ – область окна;

– окно Хемминга:

$$w(x) = 0.54 - 0.46\cos\left(\frac{2\pi x}{X}\right), \qquad (4.39)$$

где X – длина интервала $[0, x_{\max}]$, в котором задано окно, значения, не входящие в $[0, x_{\max}]$, нулевые;

– окно Блекмана:

$$w(x) = 0.42 - 0.5\cos\left(\frac{2\pi x}{X}\right) + 0.08\cos\left(\frac{4\pi x}{X}\right), \qquad (4.40)$$

– окно Ханна:

$$w(x) = 0.5\left(1 - \cos\left(\frac{2\pi x}{X}\right)\right). \qquad (4.41)$$

4.8. О фурье-преобразовании многомерных сигналов

Для преобразования многомерных сигналов необходимо полностью или фрагментарно отобразить одномерные сигналы на одномерное пространство. Рассмотрим более подробно вариант фурье-преобразования двумерного сигнала, применяемого, например, для обработки цифровых графических изображений. Цифровой образ графического изображения рассмотрим в виде матрицы чисел (скалярного поля, каждый элемент – тензор нулевого ранга) размером $X \times Y$. В случае полноцветного изображения (когда каждый элемент является вектором, т.е. тензором первого ранга) преобразование необходимо выполнять, рассматривая тензорное поле

первого ранга как совокупность тензорных полей нулевого ранга. Для матрицы скалярного поля изображения:

$$\begin{cases} f[j,i], \\ 0 \le i \le X-1, \\ 0 \le j \le Y-1 \end{cases} \tag{4.42}$$

произведём разделение на векторы. Полученные векторы длиной Y (содержащие члены ряда Фурье в виде комплексных чисел) подвергнем фурье-декомпозиции. Далее выполним конкатенацию полученных векторов так, чтобы при движении к центру вектора частота повышалась, результирующий вектор, состоящий из векторов реальной и мнимой части, запишем в $((X \times Y)$-матрицу $P[j,i]$. Затем матрицу $P[j,i]$ разделим на векторы длиной X, с последующей фурье-декомпозицией, конкатенацией и записью результата в массив $F[j,i]$.

Характерной особенностью рассмотренного преобразования является его коммутативность, т.е. последовательность преобразований сначала по строкам, а затем по столбцам эквивалентна последовательности сначала по столбцам, а затем по строкам. В большинстве случаев при записи спектральных составляющих в массивы высокочастотные составляющие располагают ближе к центру $P[j,i]$ и $F[j,i]$, а низкочастотные – на периферии, возможно и обратное расположение, но выбранного порядка необходимо придерживаться в процессе всего преобразования (декомпозиции и синтеза).

Последовательное выполнение указанных преобразований обеспечивает декомпозицию двумерного сигнала $f[j,i]$ на волны, образованные гармоническими функциями (sin и cos), характеризуемые частотой, амплитудой и вектором направления (рис. 4.3).

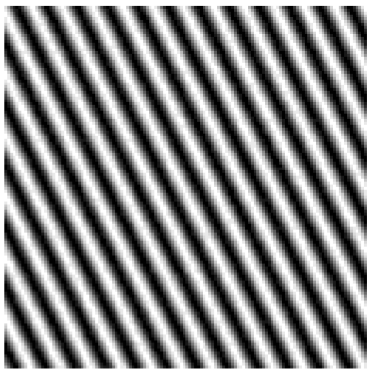


Рис. 4.3. Элемент декомпозиции двумерного сигнала

Рассмотрим пример реализации декомпозиции Фурье двумерного цифрового сигнала (листинг 4.3). Программа MATLAB позволяет производить декомпозицию и синтез изображения, а также фильтрацию (подавление) некоторого диапазона выделенных частот и визуализацию результатов

(высокочастотные составляющие располагаются ближе к центру, низкочастотные – на периферии, нумерация элементов массивов в MATLAB начинается с единицы).

Листинг 4.3. Фурье-декомпозиция цифрового изображения (MATLAB)

```
%=================================================================
clear all;       % удаление переменных
close all;       % закрытие всех окон
clc;             % очистка командного окна

% загрузка изображения
imgRGB = imread('image.bmp');
% получение размера изображения
sz=size(imgRGB);
sy=sz(1);  % высота
sx=sz(2);  % ширина
lp=10;     % параметр управления фильтрацией
% преобразование изображения в градации серого
imgGray=im2double(rgb2gray(imgRGB));
% последовательное транспонирование матрицы изображения
% и преобразование Фурье
% сначала в горизонтальном, а затем в вертикальном направлении
RF=fft(fft(imgGray).').';
% или в виде одной функции y=fft2(imgGray);
% фильтр производит подавление выделенных частот
RF((lp+1):(sy-lp),(lp+1):(sx-lp))=0;
% обратное преобразование
imgRes=ifft(ifft(RF).').';
% или в виде одной функции imgRes=ifft2(RF);
% визуализация результатов
% исходное изображение
figure('Name','Original image');
colormap(Gray); imagesc(imgGray);
% вещественная часть спектра
figure('Name','Real part');
colormap(Gray); imagesc(real(RF));
% мнимая часть спектра
figure('Name','Imaginary part');
colormap(Gray); imagesc(imag(RF));
% амплитуда
figure('Name','Amplitude');
colormap(Gray); imagesc(abs(RF));
% фаза
figure('Name','Phase');
colormap(Gray); imagesc(angle(RF));
% результат синтеза изображения
figure('Name','Synthesis image');
colormap(Gray); imagesc(abs(imgRes));
% запись в файл
imwrite(mat2gray(abs(imgRes)),'imageRes.bmp','bmp');
%=================================================================
```

Результаты работы программы на примере стандартных тестовых изображений [171] приведены на рис. 4.4, иллюстрирующем помимо специфики фурье-преобразования возможности исключения некоторой информации, например, с целью последующего сжатия.

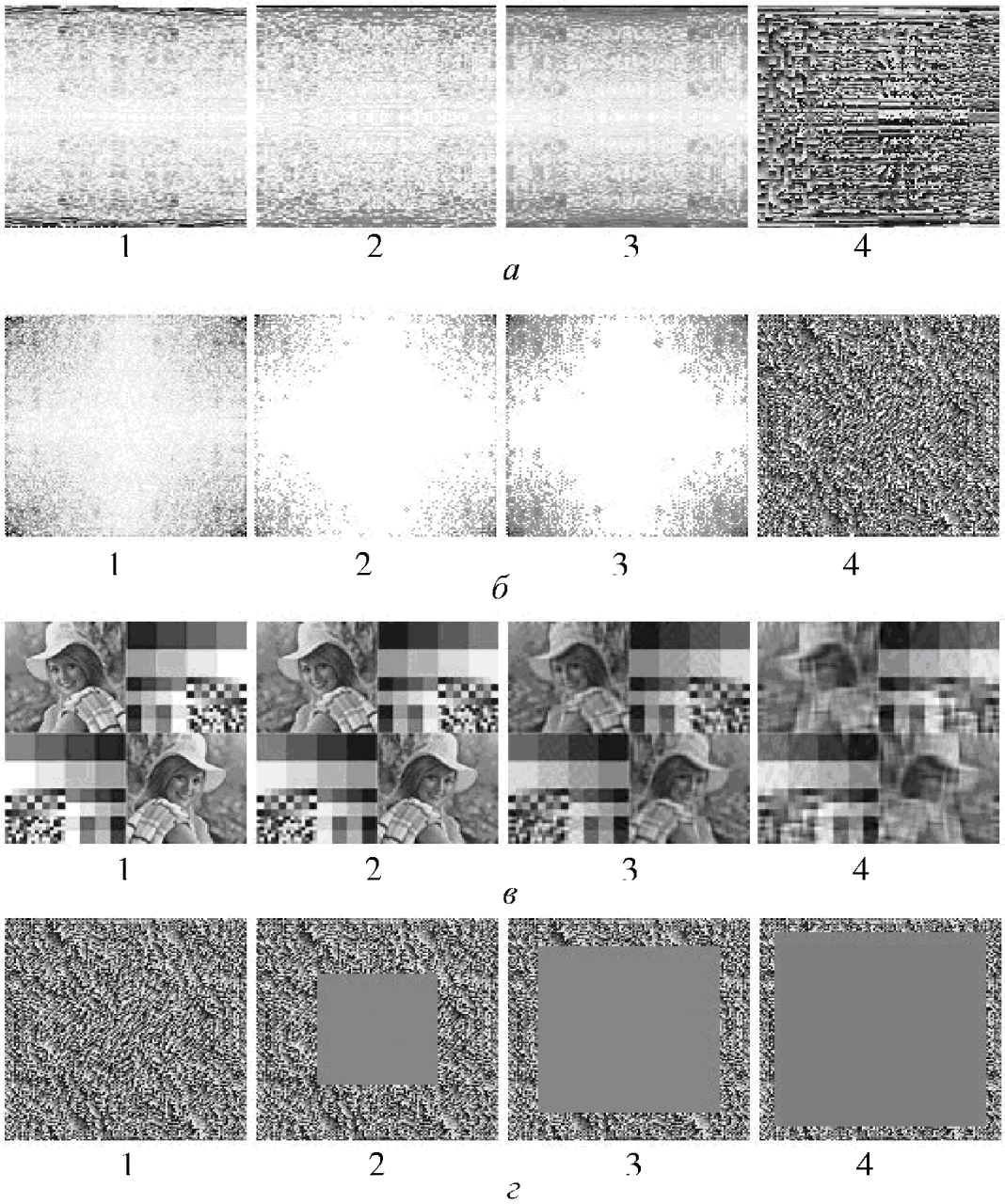


Рис. 4.4. Фурье-преобразование (*а*) столбцов матрицы изображения $f[j,i]$ и (*б*) строк матрицы $P[j,i]$: 1 – мнимая часть, 2 – реальная часть, 3 – амплитуда, 4 – фаза спектра; *в* – восстановленное изображение; *г* – подавленные элементы спектра изображения на примере фазы $F[j,i]$ (1 – сохранено 100, 2 – 75, 3 – 44, 4 – 12 % элементов спектра)

Рассмотренный принцип последовательных преобразований Фурье можно применять для сигналов любой размерности, существенным достоинством такого подхода является возможность распараллеливания процесса преобразования, что очень актуально для современных многопоточных вычислительных систем. В работе [112] подробно рассмотрено, как с помощью многомерного фурье-преобразования можно осуществить быструю свёртку и корреляцию многомерных сигналов, что может быть полезно для проблемы распознавания образов.

Рассмотрим стеганографическое встраивание информации в частотную область фурье-образа графического сигнала. В качестве контейнера используем стандартное тестовое изображение [178], уменьшенное до размеров 128×128 («128.bmp»), встраивать будем тестовое изображение, уменьшенное до 64×64 («64.bmp»). Все преобразования выполнены в программе MATLAB (листинг 4.4). В начале программы происходит загрузка исходных изображений: контейнера (рис. 4.5, *а*) и встраиваемого (рис. 4.6, *а*). Затем выполняется их фурье-преобразование и встраивание спектра изображения (рис. 4.6, *а*) в спектр (рис. 4.5, *а*). Встраивание происходит в область высоких частот. На рис. 4.7 приведены спектр контейнера до (*а*) и после встраивания (*б*), а также спектр контейнера и встроенного изображения с искажениями (*в*). Искажения возникают в результате преобразования спектра заполненного контейнера в изображение при отсечении мнимой части. Следующим действием в программе является сохранение контейнера в виде изображения с последующим чтением и извлечением содержимого (рис. 4.6, *б*). В завершение происходит визуализация и сохранение результатов в файл.

Листинг 4.4. Встраивание информации в частотную область

```
%================================================================
clear all;       % удаление переменных
close all;       % закрытие всех окон
clc;             % очистка командного окна

% загрузка исходного изображения контейнера
imgRGBcont = imread('128.bmp');
% загрузка изображения для встраивания
imgRGBemb  = imread('64.bmp');

% получение размера изображений
sz=size(imgRGBcont); % контейнер
syc=sz(1);  % высота
sxc=sz(2);  % ширина
sz=size(imgRGBemb); % встраиваемое
sye=sz(1);  % высота
sxe=sz(2);  % ширина

% фурье-образ контейнера
```

```
RFC = fft2(rgb2gray(imgRGBcont));
% фурье-образ встраиваемого изображения
RFE = fft2(rgb2gray(imgRGBemb));

% встраивание
RFC(((sxc-sxe+2)/2):((sxc+sxe)/2),((syc-sye+2)/2):((syc+sye)/2))  =
...
  RFE((1):(sxe),(1):(sye));
% обратное преобразование
imgRes=(ifft2(RFC)); % возникновение мнимой части в сигнале изо-
бражения
imgRes=real(imgRes); % потеря информации (отсечение мнимой части)

% сохранение изображения заполненного контейнера
imwrite(mat2gray(abs(imgRes)),'imageRes.bmp','bmp');
% загрузка изображения заполненного контейнера
imgRGBsteg = imread('imageRes.bmp');
% Фурье-образ изображения заполненного контейнера
RFS = fft2(imgRGBsteg);
% извлечение содержимого контейнера
imgExtract = ...
  ifft2(RFS(((sxc-sxe+2)/2):((sxc+sxe)/2),((syc-
sye+2)/2):((syc+sye)/2)));
% восстановление встроенного изображения
imgCont = ifft2(RFS);

% визуализация результатов
% исходное изображение (контейнер)
figure('Name','Original image container (empty)');
colormap(Gray);
imagesc(imgRGBcont);
% результат встраивания (заполненный контейнер)
figure('Name','Container full');
colormap(Gray);
imagesc(abs(imgCont));
imwrite(mat2gray(abs(imgCont)),'imageRes.bmp','bmp');
% амплитуда спектра заполненного контейнера
figure('Name','Amplitude');
colormap(Gray);
imagesc(log(abs(RFC)));
imwrite(mat2gray(log(abs(RFC))),'imageAmp.bmp','bmp');
% амплитуда спектра заполненного контейнера после чтения-записи
figure('Name','Amplitude rewrite');
% %colormap(Gray);
imagesc(log(abs(RFS)));
imwrite(mat2gray(log(abs(RFS))),'imageAmp_rw.bmp','bmp');
% результат извлечения
figure('Name','Extract image');
colormap(Gray);
imagesc(abs(imgExtract));
imwrite(mat2gray(abs(imgExtract)),'extract.bmp','bmp');
%================================================================
```

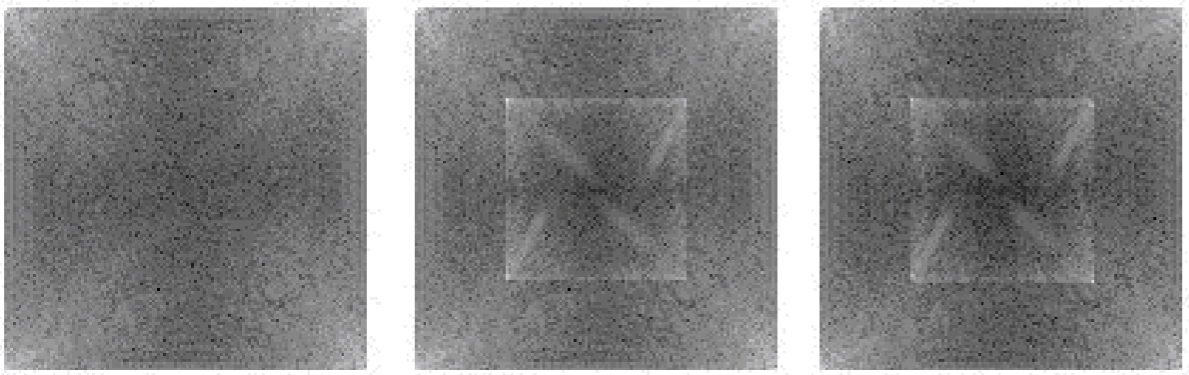
а *б*

Рис. 4.5. Контейнер:
а – исходный вид; *б* – со встроенным изображением

На рис. 4.5, *б* видна высокочастотная рябь. Сравнив рис. 4.6, *а* и *б*, можно заметить искажения во встраиваемом изображении, вызванные отсечением мнимой части при обратном фурье-преобразовании заполненного контейнера и потерей информации о фазе гармонических составляющих. Избежать потери информации позволит метод дискретного косинусного преобразования, используемого для форматов JPEG, MPEG.

а *б*

Рис. 4.6. Встраиваемое изображение:
а – исходное; *б* – извлечённое из контейнера



а *б* *в*

Рис. 4.7. Фурье-спектр:
а – исходного изображения; *б* – контейнера со встроенным изображением;
в – контейнера со встроенным изображением и частичными искажениями

4.9. Свёртка как результат произведения спектров сигналов

Одно из важных свойств свёртки – возможность её выполнения с помощью операции произведения спектра исходного сигнала $f[R^n]$ и спектра импульсной характеристики $h[R^n]$:

$$u[R^n] = f[R^n] * h[R^n] = F^{-1}\{F\{f[R^n]\} \times H\{h[R^n]\}\}, \qquad (4.43)$$

где F и H – прямое фурье-преобразование сигнала $f[R^n]$ и импульсной характеристики $h[R^n]$ соответственно; $\times$ – операция произведения спектров; F^{-1} – оператор обратного преобразования Фурье.

Произведение спектров предполагает следующие вычисления:

$$\begin{cases} \mathrm{Re}(u[R^n]) = \mathrm{Re}(F\{f[R^n]\}) \times \mathrm{Re}(H\{h[R^n]\}) - \\ \qquad - \mathrm{Im}(F\{f[R^n]\}) \times \mathrm{Im}(H\{h[R^n]\}), \\ \mathrm{Im}(u[R^n]) = \mathrm{Im}(F\{f[R^n]\}) \times \mathrm{Re}(H\{h[R^n]\}) + \\ \qquad + \mathrm{Re}(F\{f[R^n]\}) \times \mathrm{Im}(H\{h[R^n]\}), \end{cases} \qquad (4.44)$$

где Re и Im – вещественные и мнимые составляющие.

Фундаментальное свойство (4.44) существенно расширяет область применения БПФ и свёртки. Свёртку с применением БПФ и перемножением спектров называют *быстрой,* особенно актуально ее применение для обработки многомерных сигналов, когда критично время выполнения операции.

4.10. Дискретное косинусное преобразование

Дискретное косинусное преобразование (ДКП) – один из стандартных видов частотной декомпозиции по базису гармонических функций. В отличие от преобразования Фурье, в ДКП используются только косинусные функции и все преобразования происходят в области вещественных чисел, что возможно за счёт искусственного симметрирования сигнала, при котором мнимая (синусная) часть спектра обращается в нуль [112]. Для одномерного ДКП набор базисных функций описывается уравнениями вида:

$$b[x] = \cos\left[\frac{(2x+1)k\pi}{2X}\right], \qquad (4.45)$$

где $k = 0,..., K-1$ – частотный коэффициент базисных составляющих, X – размер (число дискретных элементов) вектора, x – номер отсчёта $x = 0,..., X-1$.

Одномерное ДКП можно выполнить в соответствии с выражением:

$$F[k] = w(k)\sum_{x=0}^{X-1} f[x]\cos\left[\frac{(2x+1)k\pi}{2X}\right], \qquad (4.46)$$

где $f[x]$ – элемент предназначенного для ДКП вектора размером X отсчётов, функция $w(k)$ может принимать следующие значения:

$$w(k) = \begin{cases} \sqrt{\frac{1}{X}}, & k = 0, \\ \sqrt{\frac{2}{X}}, & 1 \le k \le (K-1). \end{cases}$$

Двумерное ДКП преимущественно используется в алгоритмах сжатия статических (JPEG) и динамических изображений (MPEG). Основой такого ДКП является совокупность базисных функций, описываемых уравнением:

$$b[x, y] = \cos\left[\frac{(2x+1)k\pi}{2X}\right]\cos\left[\frac{(2y+1)p\pi}{2Y}\right], \qquad (4.47)$$

где k, p – частотные коэффициенты базисных составляющих, X, Y – размеры преобразуемого пространства. Наиболее распространённым решением является разделение пространства изображения $f[...]$ на блоки размером 8×8, 16×16 и 32×32 пикселов и выполнение ДКП для этих блоков в отдельности. Например, для случая разделения изображения на блоки 8×8 элементы $X = 8$ и $Y = 8$. При этом x и y пробегают возможные значения $x, y = 0$–7. Поскольку в большинстве случаев размеры изображения значительно превосходят размеры блоков $(X \times Y)$, ДКП изображений фактически является одной из разновидностей многомерного оконного преобразования.

Двумерное ДКП возможно осуществить следующим образом:

$$F[k, p] = w(k, p)\sum_{x=0}^{X-1}\sum_{y=0}^{Y-1} f[x, y]\cos\left[\frac{(2x+1)k\pi}{2X}\right]\cos\left[\frac{(2x+1)k\pi}{2X}\right], \qquad (4.48)$$

где $w(k, p) = \begin{cases} \sqrt{\frac{1}{XY}}, & k, p = 0, \\ \frac{2}{\sqrt{XY}}, & k, p \ge 1. \end{cases}$

На рис. 4.8 графически представлены базисные функции двумерного ДКП для блока размером 8×8, цифры отражают распространённую конфигурацию записи коэффициентов, полученных в результате прямого ДКП. Очевидно, что с увеличением значений k, p возрастает частота. На рисунке приведены оси x и y, в элементе с координатами $x = 0$, $y = 0$ сохраняется

значение постоянной составляющей, а в элементе $x=7,\ y=7$ – значение составляющей, содержащей наибольшую частоту по двум направлениям.

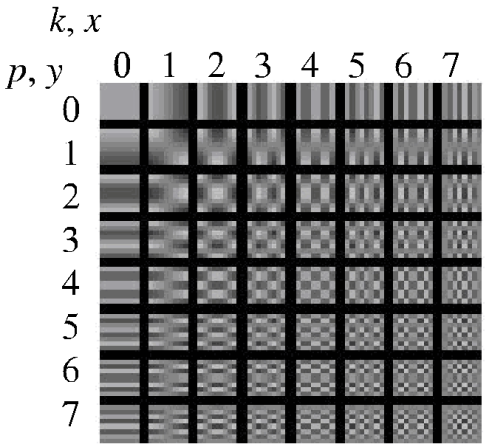


Рис. 4.8. Графическое представление базисных функций двумерного ДКП

4.11. Вейвлет-преобразование

Во многих физических процессах наблюдаются уединённые волны, или солитоны [1, 2, 68, 157]. Математическим аналогом солитона является *вейвлет* (от англ. *wavelet* – всплеск), в отличие от волн, имеющих в общем случае неограниченную протяжённость, протяжённость вейвлета ограничена.

Рассмотрим *вейвлет-преобразование* [123, 126]. Вейвлетом называют (действительную или комплексную) функцию $g(x)$:

$$\int_{-\infty}^{+\infty} g(x)dx = \text{const}, \tag{4.49}$$

быстро затухающую при $x \to \pm\infty$. Наиболее распространены вейвлеты, для которых выполняется условие $\int_{-\infty}^{+\infty} g(x)dx = 0$, такие функции осциллируют относительно оси x и вклады в интегральную сумму положительных и отрицательных значений компенсируют друг друга.

Вейвлет должен обладать условием допустимости, т.е. характеризоваться конечной энергией:

$$C_g = 2\pi \int_{-\infty}^{+\infty} \frac{|g(\omega)|^2}{|\omega|} d\omega < \infty, \tag{4.50}$$

где ω – собственная частота вейвлета.

На базе $g(x)$ получают семейство функций:

$$g(x,q,u)=\frac{1}{\sqrt{q}}g\left(\frac{x-u}{q}\right), \tag{4.51}$$

где u определяет сдвиг функции $g(x)$, q определяет масштабирование: $q<1$ – сжатие, $q>1$ – растяжение. Функцию $g(x)$, порождающую семейство $g(x,q,u)$, при $q=1$ и $u=0$ принято называть *материнским вейвлетом.*

На основе семейства функций (4.51) строится интегральное вейвлет-преобразование:

$$F(q,u)=\frac{1}{\sqrt{C_g}}\frac{1}{\sqrt{q}}\int_{-\infty}^{+\infty}f(x)\overline{g}\left(\frac{x-u}{q}\right)dx, \tag{4.52}$$

т.е. выполняется отображение исходного сигнала $f(x)$ в фазовое пространство, это позволяет выполнить оконное преобразование Фурье, которое можно рассматривать как разновидность вейвлет-преобразования.

Для ортогональных и биортогональных вейвлет-преобразований справедлив аналог теоремы Парсеваля о равенстве энергии сигнала, выраженного в частотной или пространственной области:

$$\int_{-\infty}^{+\infty}\left|f(x)\right|^2dx=C_g^{-1}\int_0^{\infty}\int_{-\infty}^{+\infty}\frac{\left|F(q,u)\right|^2}{q^2}dudq. \tag{4.53}$$

В листинге 4.5 приведён пример программы вейвлет-преобразования сигнала в среде MATLAB, в качестве материнского использован вейвлет «мексиканская шляпа» (Mexican Hat Wavelet, mexh). Результатом исполнения программы является диаграмма (рис. 4.9), соответствующая распределению энергии вейвлетов.

Листинг 4.5. Вейвлет-преобразование (MATLAB)

```
%================================================================
clear all;                          % удаление переменных
close all;                          % закрытие всех окон
clc;                                % очистка командного окна
% задание параметров и сигнала для преобразования
size=1024;                          % длина
step=(2*pi)/size;                   % шаг
x=-pi:step:(pi-step);               % вектор аргумента
y=10*sin(4*x)+5*sin(32*x);          % вектор значений
% вейвлет-преобразование
scales = 0.1:0.1:22;                % вектор значений масштабирующей пе-
ременной
coefs = cwt(y,scales,'mexh'); % расчёт вейвлет-коэффициентов
% визуализация результатов
figure('Name','Result window');
```

```
colormap(gray(64)); % палитра: оттенки серого
wscalogram('image',coefs,'scales',scales,'ydata',y);
%=================================================================
```

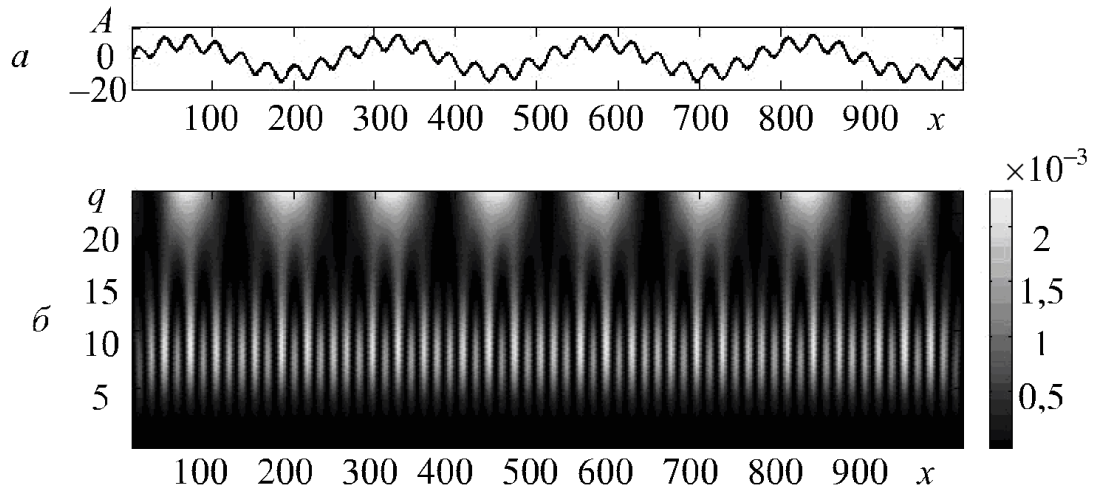


Рис. 4.9. Результат выполнения программы (листинг 4.5).
а – график исходного сигнала; *б* – распределение энергии вейвлет-коэффициентов, т.е. отображение сигнала в фазовое пространство (*q* – масштабный коэффициент вейвлета, *x* – номер отсчёта)

Вейвлет-преобразование – эффективный инструмент обработки и анализа различных сигналов, используемый во многих научно-практических приложениях. Особенности сигнала могут быть учтены на этапе выбора материнского вейвлета, что способствует значительному повышению эффективности преобразования. Однако бывает затруднительно выбрать материнский вейвлет, подходящий для преобразования различающихся сигналов, объединённых в класс по ряду признаков (например, фотографии, видео, данные медицинских наблюдений и т.д.), поэтому для обобщения класса преобразований над классом сигналов формируют библиотеки вейвлетов [123], что является достаточно трудоёмкой задачей, особенно для отнесённых к одному классу многомерных сигналов, имеющих существенные различия.

4.12. Преобразование Лапласа

Множество реальных физических процессов описывается гармоническими и экспоненциальными функциями. Поэтому существенное значение для математического описания реальных систем имеет *преобразование Лапласа*[10]. Двустороннее преобразование Лапласа позволяет получить *изо-*

[10] Пьер-Симон Лаплас (23 марта 1749, Кальвадос – 5 марта 1827, Париж) – математик, физик и астроном; известен работами в области небесной механики, дифференциальных уравнений, один из создателей теории вероятностей.

бражение $F(p)$ непрерывной функции $f(x)$ в интервале $(-\infty;+\infty)$ на *S*-плоскости:

$$F(p)=\int_{-\infty}^{+\infty} f(x)e^{-px}dx\,, \tag{4.54}$$

где $p=\sigma+j\omega$ – комплексный аргумент изображения, называемый *оператором*, σ – действительное число, ω – круговая частота. Функция $f(x)$ должна удовлетворять условиям Дирихле, т.е. за любой конечный промежуток иметь конечное число разрывов первого рода и конечное число максимумов и минимумов. Смысл преобразования Лапласа позволяет раскрыть выражение (4.54) в виде:

$$F(\sigma,\omega)=\int_{-\infty}^{+\infty} f(x)e^{-\sigma x}\ e^{-j\omega x}dx\,. \tag{4.55}$$

Исходный сигнал $f(x)$ умножается на экспоненциальный множитель $e^{-\sigma x}$, в результате получается выражение $f(x)e^{-\sigma x}$, умножение полученного выражения на $e^{-j\omega x}$ с последующим интегрированием:

$$V(\omega)=\int_{-\infty}^{+\infty} v(x)e^{-j\omega x}dx \tag{4.56}$$

позволяет получить комплексное фурье-преобразование сигнала вида $v(x)=f(x)e^{-\sigma x}$.

Таким образом, преобразование Лапласа является совокупностью преобразований Фурье для произведений исходного сигнала $f(x)$ на экспоненциальный множитель $e^{-\sigma x}$. Пространственно преобразование Лапласа можно представить в виде тензорного поля на *S*-плоскости, заданной комплексными координатами $(\sigma, j\omega)$, каждая точка которого соответствует некоторой комплексной величине $F(\sigma,\omega)$.

Обратное преобразование Лапласа имеет следующий вид:

$$f(x)=\frac{1}{2\pi j}\int_{\sigma_0-i\infty}^{\sigma_0+i\infty} F(p)e^{px}dp\,, \tag{4.57}$$

где $p=\sigma_0+j\omega$.

Преобразование Лапласа позволяет с помощью операторного отображения переходить от дифференциальных и интегральных уравнений к

обыкновенным алгебраическим [60, 64, 69]. Рассмотрим односторонний случай, удовлетворяющий казуальным системам, производная $f'(x) = \dfrac{df(x)}{dx}$ будет иметь вид:

$$L\ f'(x)\ = \int_0^{\infty} f'(x)e^{-px}dx\,, \tag{4.58}$$

здесь $L\ f'(x)$ обозначает преобразование Лапласа функции $f'(x)$.

Проинтегрировав выражение (4.58) по частям, получим:

$$\int_0^{\infty} f'(x)e^{-px}dx = e^{-px}f(x)\Big|_0^{\infty} + p\int_0^{\infty} e^{-px}f(x)dx = pF(p) - f(0)\,. \tag{4.59}$$

Для производной высших порядков в общем случае можно записать:

$$\begin{aligned} L\ f^{(n)}(x)\ &= p^nF(p) - p^{n-1}f(0) - p^{n-2}f'(0) - \\ &- p^{n-3}f''(0) - \ldots - f^{(n-1)}(0), \end{aligned} \tag{4.60}$$

для случая $f(0) = 0$:

$$L\ f^{(n)}(x)\ = p^nF(p)\,. \tag{4.61}$$

Отметим, что в выражении (4.61) слагаемое $pF(p)$ соответствует широко применяющемуся в электротехнике преобразованию Карсона–Хевисайда:

$$K(p) = pL\{f(x)\} = p\int_0^{+\infty} f(x)e^{-px}dx\,, \tag{4.62}$$

особенность которого – одинаковая размерность изображения и оригинала.

В одностороннем случае $\varphi(x) = \int_0^x f(x)dx$ имеет следующий вид:

$$\begin{aligned} L\left\{\int_0^x f(x)dx\right\} &= \int_0^{\infty}\varphi(x)e^{-px}dx = \\ &= -\varphi(x)\frac{1}{p}e^{-px}\Big|_0^{\infty} + \frac{1}{p}\int_0^{\infty}\varphi'(x)e^{-px}dx. \end{aligned} \tag{4.63}$$

С учётом того, что функция $f(0)=0$ и $f(\infty)=0$, первое слагаемое (4.63) $-\varphi(x)\frac{1}{p}e^{-px}\Big|_0^\infty = 0$, а второе $\frac{1}{p}\int_0^\infty \varphi'(x)e^{-px}dx = \frac{1}{p}\int_0^\infty f(x)e^{-px}dx = \frac{1}{p}F(p)$, окончательно изображение интеграла будет иметь вид:

$$L\left\{\int_0^x f(x)dx\right\} = \frac{F(p)}{p}. \qquad (4.64)$$

Преобразование Лапласа широко применяется при решении задач электротехники, ядерной физики, механики, сопротивления материалов и др.

В результате преобразования Лапласа импульсной характеристики можно получить полную характеристику системы, исследуя только *нули,* т.е. точки, где $F(\sigma,\omega)=0$, и *полюсы* – точки, где $F(\sigma,\omega)=\infty$.

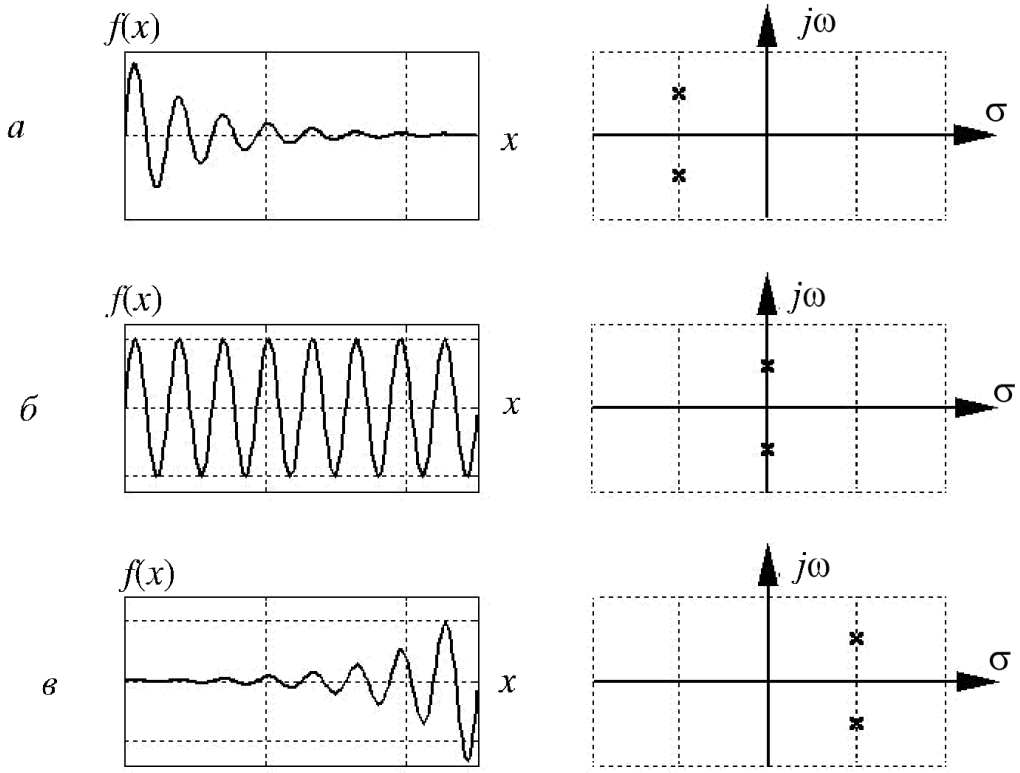


Рис. 4.10. Графические образы исходных функций $f(x)$ и их полюсы, построенные на *S*-плоскости

Одной из важнейших характеристик любой системы является устойчивость. Систему можно считать:

– устойчивой, когда её полюсы расположены в левой *S*-полуплоскости (рис. 4.10, *а*) $f(x)=e^{-x}\sin(x)$, $L\{f(x)\}=F(p)=\frac{1}{p^2+2p+2}$;

– условно устойчивой, когда полюсы расположены на оси $j\omega$ ($\sigma=0$) (рис. 4.10, *б*) $f(x)=\sin(x)$, $L\{f(x)\}=F(p)=\dfrac{1}{p^2+1}$;

– неустойчивой, если полюсы расположены в правой *S*-полуплоскости (рис. 4.10, *в*) $f(x)=e^x\sin(x)$, $L\{f(x)\}=F(p)=\dfrac{1}{p^2-2p+2}$.

Устойчивость многополюсной системы определяется положением полюса, имеющего наибольшее значение σ. Таким образом, критерием устойчивости системы является положение всех полюсов в левой полуплоскости.

4.13. Z-преобразование

Z-преобразование позволяет получить отображение дискретного сигнала $f[x]$ на *Z*-плоскость. Дискретное *Z*-преобразование описывается выражением:

$$F(z)=\sum_{x=-\infty}^{+\infty} f[n]z^{-x}, \qquad (4.65)$$

где $z=re^{j\omega}$. *Z*-преобразование, так же как и преобразование Лапласа, позволяет исследовать отклик системы с импульсной характеристикой $f[n]$ на гармонические осцилляции, нарастающие и затухающие экспоненциально с различными скоростями.

Учитывая, что $z=re^{j\omega}$, выражение (4.65) можно записать в виде:

$$F(z)=\sum_{x=-\infty}^{+\infty} f[n]re^{-j\omega x}=\sum_{x=-\infty}^{+\infty}(f[n]r^{-x})e^{-j\omega x}, \qquad (4.66)$$

таким образом, *Z*-преобразование можно рассматривать как совокупность преобразований Фурье для произведений некоторого дискретного сигнала $f[n]$ на экспоненциальный множитель r^{-x}.

На рис. 4.11 приведены примеры исходных импульсных характеристик систем $f[n]$ и положения полюсов и нулей *Z*-преобразований. Критерием устойчивости системы является расположение всех полюсов, полученных в результате *Z*-преобразования импульсной характеристики, внутри окружности единичного радиуса, расположение полюсов на линии окружности единичного радиуса является признаком условно устойчивой системы, в ином случае система считается неустойчивой.

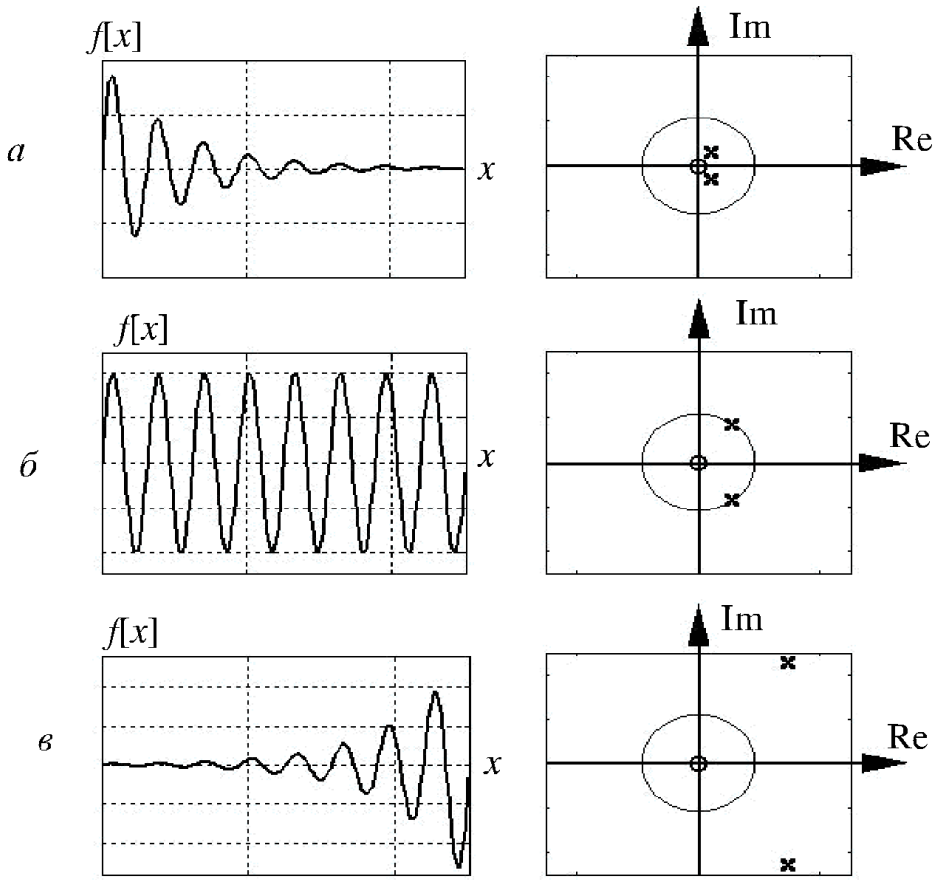


Рис. 4.11. Графические образы исходных функций $f[x]$:
система *а* – устойчива, *б* – условно устойчива (генератор), *в* – неустойчива

Для удобства сравнения характеристик различных фильтров используют определяемую согласно теореме Котельникова нормированную частоту $\hat{f}$, позволяющую привести спектр рассматриваемого сигнала к основной полосе частот $\left[0;\frac{f_д}{2}\right]$:

$$\hat{f}=\frac{f}{f_д}=fT \text{ или } \hat{\omega}=\frac{\omega}{f_д}=\omega T, \qquad (4.67)$$

$f_д$ – частота дискретизации; f – реальная частота; $\omega=2\pi f$ – реальная круговая частота. Таким образом, основная полоса частот сводится к диапазону $\hat{f}\in 0;\ 0.5$ и $\hat{\omega}\in 0;\ \pi$ [108]. На Z-плоскости положение точки определяется углом поворота ω относительно начала координат и положительного направления вещественной оси, причём обычно используют нормированную круговую частоту $\hat{\omega}$, таким образом ограничивают угол поворота $\hat{\omega}\in -\pi;\ \pi$, охватывающий положительные и отрицательные частоты.

Наиболее широко Z-преобразование применяется при анализе и синтезе бесконечной импульсной характеристики фильтров.

4.14. О неопределённости при спектральных преобразованиях сигналов

В 1927 г. Гейзенберг[11] выдвинул предположение, что существует фундаментальный закон природы, объясняющий некоторые, парадоксальные с точки зрения классической физики, явления квантового мира. Этот закон он назвал *принципом неопределённости*, который ограничивает применение к микрообъектам классических понятий и представлений [117].

Количественно этот принцип выражается в виде *соотношений неопределённостей*. Первое из них можно записать в виде:

$$\Delta x \Delta p_x \geq 2\pi\hbar, \qquad (4.68)$$

где Δx и Δp_x – неопределённость координаты x и проекции импульса p_x, $\hbar = 1.05457160 \cdot 10^{-34}$ Дж·с – универсальная постоянная Планка[12]. В общем случае соотношение Робертсона–Шрёдингера выражает неопределённость значений любых двух переменных, которые не коммутируют друг с другом. Под неопределённостью понимают минимально возможную погрешность результатов одновременного измерения величин, связанных соотношением (4.68), такие погрешности в квантовой механике считаются не следствием несовершенства измерительной техники, а принципиальным законом, описывающим количественный смысл соотношения неопределённостей.

В качестве иллюстрации неопределённости Гейзенберг приводил пример [152] с воображаемым микроскопом, с помощью которого исследователь измеряет положение и импульс электрона, отражающего падающей на него фотон и тем обнаруживающего своё присутствие. Энергия фотона описывается уравнением $E = \hbar\omega$, где ω – круговая частота излучения $\omega = 2\pi f$. Если фотон имеет малую длину волны $\lambda = \dfrac{2\pi c}{\omega}$ (где c – скорость света) и, следовательно большой импульс, то положение электрона может быть измерено вполне точно, при этом фотон рассеивается случайным образом, передавая электрону значительную и неопределенную долю своего импульса. Если у фотона большая длина волны и малый импульс, он мало изменяет импульс электрона, но рассеяние будет определять положение

[11] Гейзенберг Вернер Карл (5 декабря 1901, Вюрцбург – 1 февраля 1976, Мюнхен) – физик-теоретик, один из создателей квантовой механики; лауреат Нобелевской премии по физике (1932).

[12] Планк Макс Карл Эрнст Людвиг (23 апреля 1858, Киль – 4 октября 1947, Гёттинген) – физик-теоретик, основоположник квантовой теории, член Берлинской АН (1894).

электрона очень неточно. В результате произведение неопределённостей в координате и импульсе остаётся не меньшим постоянной Планка.

Гейзенберг не сформулировал точное математическое выражение для принципа неопределённости, а использовал принцип как эвристическое количественное соотношение.

Неопределенность, замеченная в явлениях квантового мира, проявляется и в других областях. В теории и практике цифровой обработки сигнала известно, что чем *уже* сигнал во временной области, тем шире его образ в частотной области, и наоборот. Вообще говоря, единичный импульс относится к классу так называемых *пар преобразований Фурье*. Если некоторому сигналу $f(x)$ во времени в частотной области соответствует образ $F\ f(x)$, то в частотной области спектр, имеющий форму сигнала $f(x)$, во временной области будет представлен образом $F\ f(x)$.

Если взглянуть более внимательно на представленный принцип дуальности пар преобразований Фурье, то можно заметить, что функция единичного импульса $\delta(x)$ при $\Delta x \to 0$ и неограниченные по протяжённости гармонические функции $F\ \delta(x)$ несоизмеримы друг с другом, как несоизмеримы, например, радиус и длина окружности (рис. 4.12, *a* – масштабный коэффициент; из рисунка видно, что выбор заданной окружности в качестве меры других окружностей исключает неопределённость, вызванную несоизмеримостью). Гармоническая функция имеет неограниченную протяжённость, в то время как исходный сигнал $\delta(x)$ конечен. Фактически эти две функции имеют различные меры, что и приводит к невозможности ограничения спектра или протяжённости, в зависимости от направления преобразования. Поскольку преобразование Фурье – это мера, построенная на корреляционных отношениях базисных гармонических функций и сигнала, применение этой меры для измерения несоизмеримых величин приведет к возникновению неопределённостей. Проявлением таких неопределённостей являются колебания Гиббса.

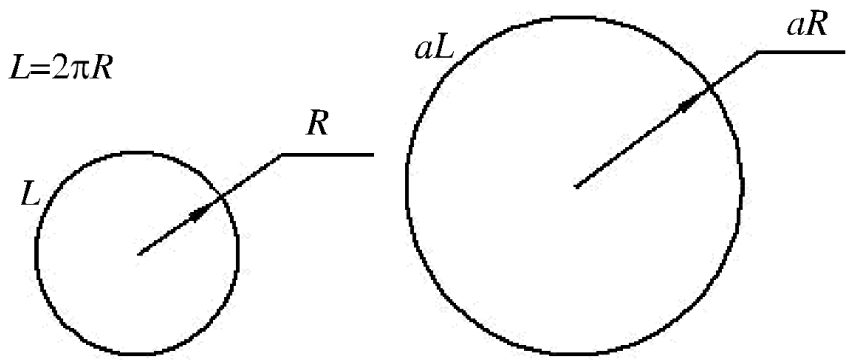


Рис. 4.12. Заданная окружность как мера

В преобразовании Лапласа используются две меры – экспоненциальная и гармоническая, формальная запись которых имеет вид (4.55); в вейвлет-преобразовании базовой мерой служит материнский вейвлет (в практических расчетах возможно выбрать наиболее подходящий). Заметим, что многие процессы в окружающем мире подчинены гармоническим и экспоненциальным законам, поэтому преобразование Лапласа находит применение во многих моделях, описывающих явления реального мира. Таким образом, выбор меры является основой для снижения неопределённости.

Согласно парадоксу Эйнштейна–Подольского–Розена (ЭПР) [143], изменение физической величины одной частицы в квантовой механике должно приводить к изменению вероятности распределения другой частицы, сцепленной с первой, причём со скоростью, которая может превышать скорость света. Такое поведение квантовых частиц согласуется с принципом Паули[13] (принципом запрета) который был сформулирован в 1925 г. для электронов и в дальнейшем распространён на все частицы с полуцелым спином. Парадокс ЭПР подтвержден экспериментально: в 1997 г. была осуществлена передача поляризационного состояния фотона [135, 136], а в 2004 г. – телепортация квантового состояния атома.

Успех эксперимента в области квантовой телепортации означает, что в качестве меры для определения состояния системы может быть использована система, связанная с ней набором квантовых чисел.

Во многих практических приложениях цифровой обработки сигналов качество результата может существенно зависеть от выбранной меры, поэтому на этапе разработки цифровых систем выбор принципов и способов преобразования, и следовательно меры, является ключевым.

[13] Паули Вольфганг Эрнст (25 апреля 1900, Вена – 15 декабря 1958, Цюрих) – физик, лауреат Нобелевской премии.

5. Способ декомпозиции *n*-мерных сигналов по базису прямоугольных всплесков

Необходимость повышения быстродействия вычислительных систем за счёт оптимизации и применения новых алгоритмов стимулирует поиски новых способов, позволяющих выполнять частотный синтез и декомпозицию. В настоящей главе рассмотрен предложенный автором способ декомпозиции цифровых сигналов по базису прямоугольных всплесков, эффективность которого подтверждена при решении ряда задач распознавания образов. По результатам разработки и исследования способа получен патент на изобретение [49].

5.1. Формальное описание способа

Способ декомпозиции [49] по *базису прямоугольных всплесков* (БПВ) ориентирован на цифровые (дискретные) сигналы [37], образованные совокупностью прямоугольников (рис. 5.1). Таким образом, наилучшей мерой для цифровых сигналов могут быть прямоугольные функции или их некоторые объединения, являющиеся разновидностью вейвлетов.

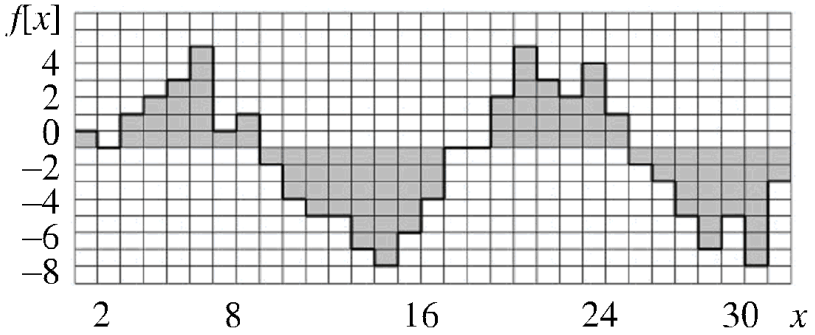


Рис. 5.1. Дискретность цифрового сигнала $f[x]$

Декомпозиция по БПВ [45] позволяет получить спектр *n*-мерного цифрового сигнала и локализовать его в пространстве и (или) времени, фактически отображая цифровой сигнал в фазовое пространство. Эта задача решается в ходе прямого преобразования (декомпозиции) путем последовательных итеративных вычислений в соответствии с выражением:

$$f_{k-1} = f_k - S_k , \qquad (5.1)$$

где S_k – спектральный компонент остаточного сигнала f_k.

Значение индекса k соответствует протяжённости (периоду) взаимно перпендикулярных и параллельных элементов спектральных компонентов, при $k = K$ сигнал f_K является исходным. Сумма полученных в ходе прямого преобразования спектральных компонентов S_k соответствует обратному преобразованию (синтезу):

$$f_K = \sum_{k=1}^{K} S_k \ . \qquad (5.2)$$

Разложение n-мерного сигнала на спектральные компоненты происходит не по выбранному заранее, а по формируемому для данного сигнала n-мерному базису (базисной функции), образованному взаимно параллельными и перпендикулярными (для $n > 1$) элементами исходного сигнала.

Рассмотрим предлагаемый способ на примере сигнала, каждый элемент которого является тензором нулевого ранга, т.е. скаляром (такой сигнал образует скалярное поле, которое является частным случаем тензорного). В качестве примера скалярных полей можно указать: поле распределения температуры, плотности, электростатического потенциала, поле яркости и т.д. Если сигнал не является скалярным полем, для дальнейшей декомпозиции его необходимо разложить на некоторое ограниченное множество скалярных полей. Как уже было показано ранее, двумерный сигнал цифрового изображения в пространстве RGB является тензорным полем, причём каждому элементу сигнала соответствует тензор первого ранга, т.е. трёхмерный вектор, образованный компонентами RGB. Такой сигнал можно рассматривать как совокупность трёх скалярных полей – по отдельности для каждого цветового компонента.

Дискретную структуру, имеющую размерность, равную размерности исходного сигнала с отличной от нуля амплитудой (высотой), будем называть *элементарным всплеском*. В направлении выделения протяжённость элементарного всплеска может иметь любое отличное от нуля значение, но не более размера исходного сигнала в этом направлении, по другим направлениям размеры элементарного всплеска равны единице дискретизации соответствующих направлений.

Разделение исходного сигнала f_k на спектральные компоненты происходит начиная с самых протяжённых элементарных всплесков. В каждой последующей итерации, согласно (5.1), протяжённость элементарного всплеска на единицу меньше, чем в предыдущей. При выделении из исходного сигнала взаимно перпендикулярных всплесков учитывается принцип суперпозиции волн, т.е. сложения их амплитуд в месте пересечения. Поэтому последовательность обхода направлений для выполнения условия однозначности разложения должна быть одинаковой при формировании каждого спектрального компонента S_k .

Преобразования многомерного и одномерного сигнала различаются тем, что в первом случае нет заранее выбранного многомерного материнского вейвлета – базисной функции [41]. Такая функция формируется из элементов исходного сигнала, отдельно для каждой спектральной составляющей, совокупность которых образует спектр сигнала (они могут обладать уникальной конфигурацией и в общем случае не выражаться через взаимные линей-

ные преобразования). В таком контексте понятие «материнский вейвлет» может быть применено по отношению к элементарному всплеску.

Приведем обобщённый алгоритм формирования спектральных составляющих (прямое преобразование) для N -мерного скалярного поля в системе координат $x_1, x_2, ..., x_N$. Используемая в алгоритме формулировка «определенность k сигнала в заданном направлении» означает, что подлежащий декомпозиции исходный сигнал имеет в этом направлении размер не менее k.

1. На поле исходного сигнала f_K выбирают начальную величину $K = \max(x_1, x_2, ..., x_N)$ как значение максимальной протяженности существующих в f_K значений сигнала (определенности) в направлениях координатных осей $x_1, x_2, ..., x_N$.

(*В последующих пунктах алгоритма последовательно в порядке обхода направлений $x_1, x_2, ..., x_N$ выделяют перпендикулярные и параллельные элементарные всплески.*)

2. Задают начальные координаты выделения из сигнала элементарных всплесков $n = 1$.

3. Проверяют определённость сигнала в выбранном направлении: если x_n не меньше k, то переход к п. 4, иначе – к п. 6.

4. Выделяют вдоль направления x_n элементарные всплески протяженностью k, всплеск должен удовлетворять следующим условиям:

– границами считаются элементы, имеющие либо нулевое значение, либо отличное его по знаку (в случае знакопеременного сигнала);

– амплитуда элементарного всплеска определяется как минимальное значение амплитуды между границами всплеска;

– единичная протяжённость по всем направлениям положения, кроме x_n.

5. Локализованные (см. п. 4) элементарные всплески вычитают из остаточного сигнала f_k и добавляют к спектральной составляющей S_k в соответствии с выражением $f_{k-1} = f_k - S_k$.

6. Если $n < N$, то $n = n + 1$ и переход к п. 3, иначе – к п. 7.

7. Если значение $k > 1$, то $k = k - 1$ и переход к п. 2, иначе – окончание алгоритма.

Рассмотренный алгоритм позволяет производить декомпозицию многомерных сигналов по базису прямоугольных всплесков.

Декомпозиция по базису прямоугольных всплесков (5.1) и синтез (восстановление) сигнала из элементов декомпозиции (5.2) образуют систему операций, называемую *преобразованием по базису прямоугольных всплесков* (далее преобразование по БПВ).

5.2. Декомпозиции для одномерного случая

Рассмотрим примеры декомпозиции одномерного сигнала. Приведенный на рис. 5.2 пример (f_{12} – исходный сигнал, f_0 – нулевой остаточный сигнал, A – амплитуда сигналов и спектральных составляющих) демонстрирует возможность применения предложенного способа:

– если положить, что ось x характеризует время, то совокупность компонентов S_4, S_3, S_2, S_1 в каждый отдельный момент времени позволяет получить спектр сигнала, при этом не нужно производить вычисления спектра в сдвигаемом окне, как это требуется при фурье-преобразовании для получения отображения сигнала на фазовую плоскость;

– используя предложенный способ, возможно выполнять декомпозицию всего сигнала или любого выбранного фрагмента, получая в результате отображение сигнала в фазовое пространство, а в случае одномерного сигнала – на фазовую плоскость (x, S) .

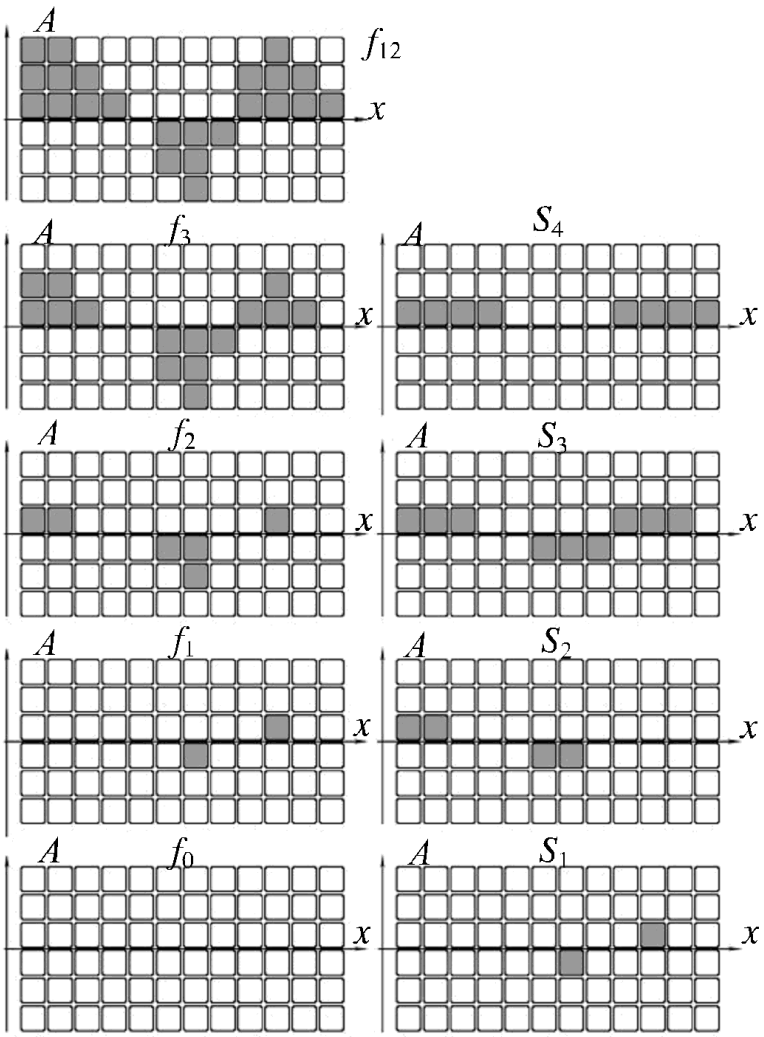


Рис. 5.2. Последовательное разложение на спектр одномерного сигнала

На рис. 5.3, 5.4 графически отображены результаты декомпозиции исходного цифрового сигнала $f[x]$ (*а*) по БПВ (*б*), а также с помощью оконного преобразования Фурье (*в*). Из рисунков видно, что рассмотренный способ (см. рис. 5.3, *б* и 5.4, *б*) позволяет выявить распределение частот в области определения функции $f[x]$. По сравнению с преобразованием Фурье локализация частот выполнена более эффективно, особенно – для меандроподобного сигнала, это обусловлено тем, что элементарные всплески имеют прямоугольную форму, следовательно, являются наилучшей мерой для таких сигналов.

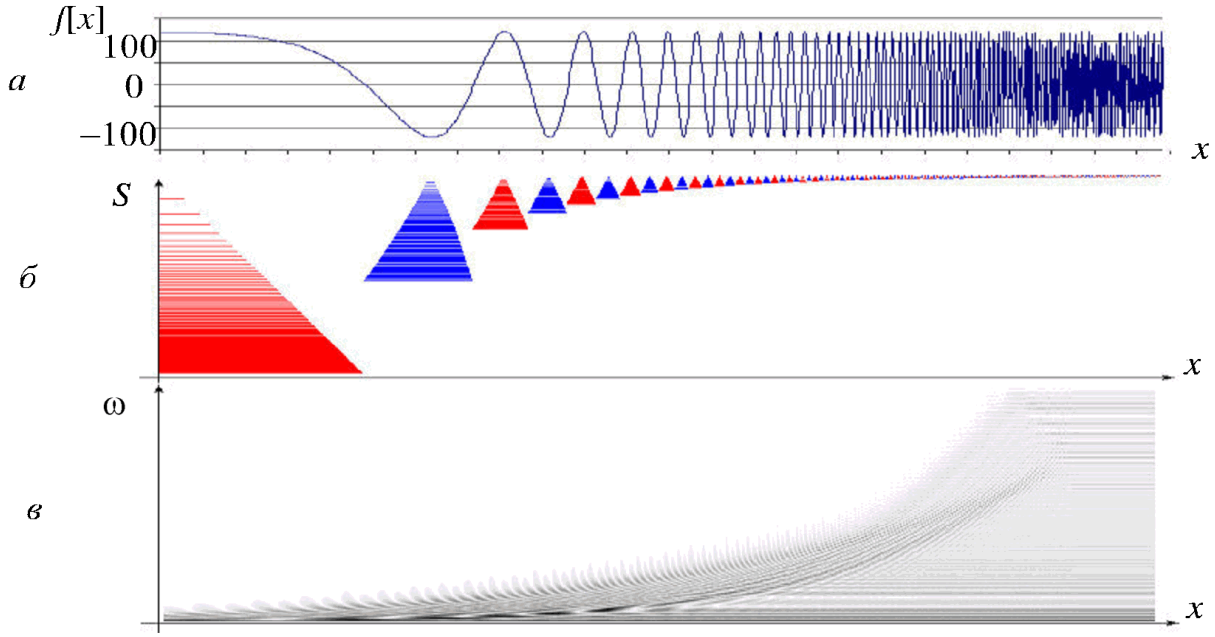


Рис. 5.3. Графическое отображение декомпозиции сигнала $f[x]$: красный цвет – всплески с положительной амплитудой, синий – с отрицательной

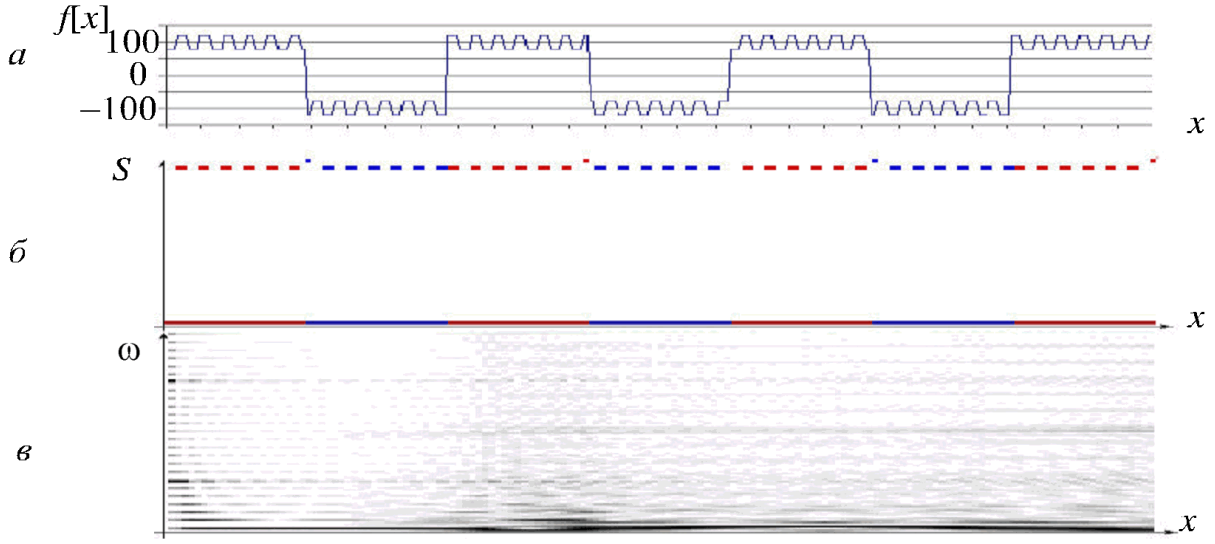


Рис. 5.4. Графическое отображение декомпозиции меандроподобного сигнала $f[x]$: красный цвет – всплески с положительной амплитудой, синий – с отрицательной

5.3. Декомпозиция двумерного сигнала

Декомпозицию многомерных сигналов можно рассмотреть на примере двумерных, поскольку двумерный случай отражает практически все особенности преобразования многомерных сигналов и достаточно удобен с точки зрения графической иллюстрации [40].

На рис. 5.5 приведен пример частичной декомпозиции: из двумерного сигнала f_{10} выделены элементарные всплески протяжённостью четыре единицы. В соответствии с принципом суперпозиции волн (сложения амплитуд в точке пересечения) выполнена декомпозиция, результат которой зависит от последовательности обхода направлений (x_1, x_2) или (x_2, x_1). Рассмотрены случаи выделения компонентов S_{4x_1} и S_{4x_2}, очевидно, что конфигурация остаточных сигналов f_{4x_1} и f_{4x_2} различается.

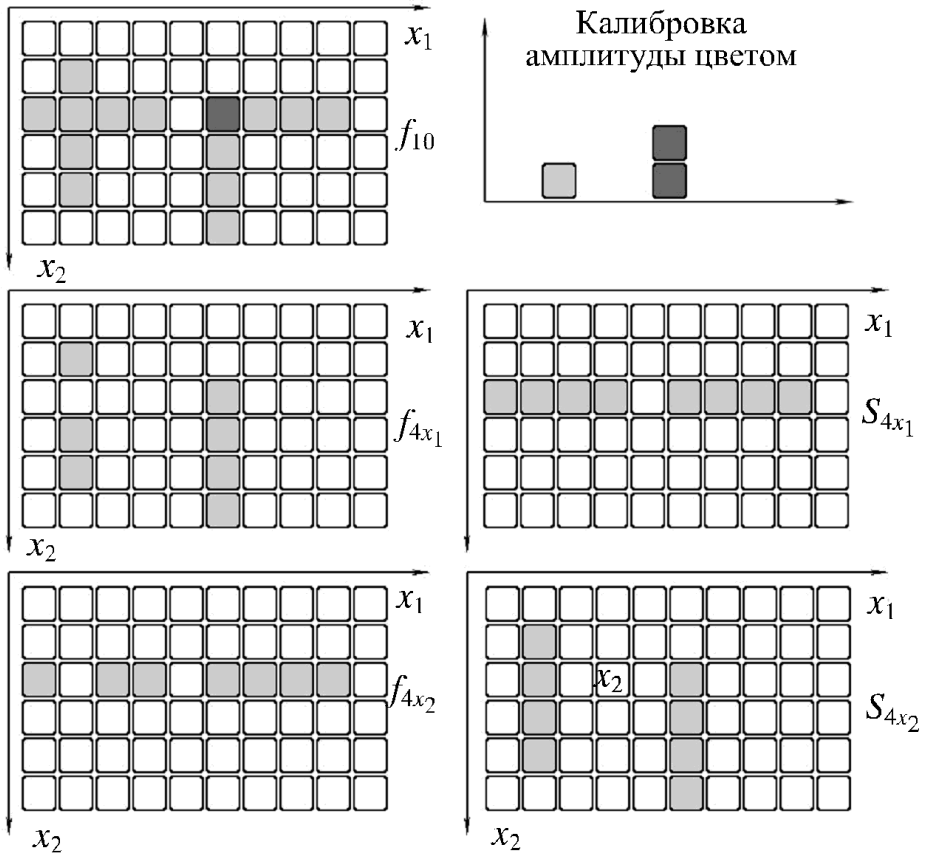


Рис. 5.5. Частичная декомпозиция двумерного сигнала f_{10} в пространстве, заданном координатами (x_1, x_2)

На рис. 5.6 представлена полная декомпозиция двумерного сигнала f_{10}, определённого в пространстве с координатами (x_1, x_2). Все спектральные компоненты по направлениям объединены в S_5, S_2, S_1 согласно кри-

терию равенства протяжённости элементарных всплесков (f_4, f_1, f_0 – элементы остаточного сигнала; S_5, S_2, S_1 – элементы декомпозиции).

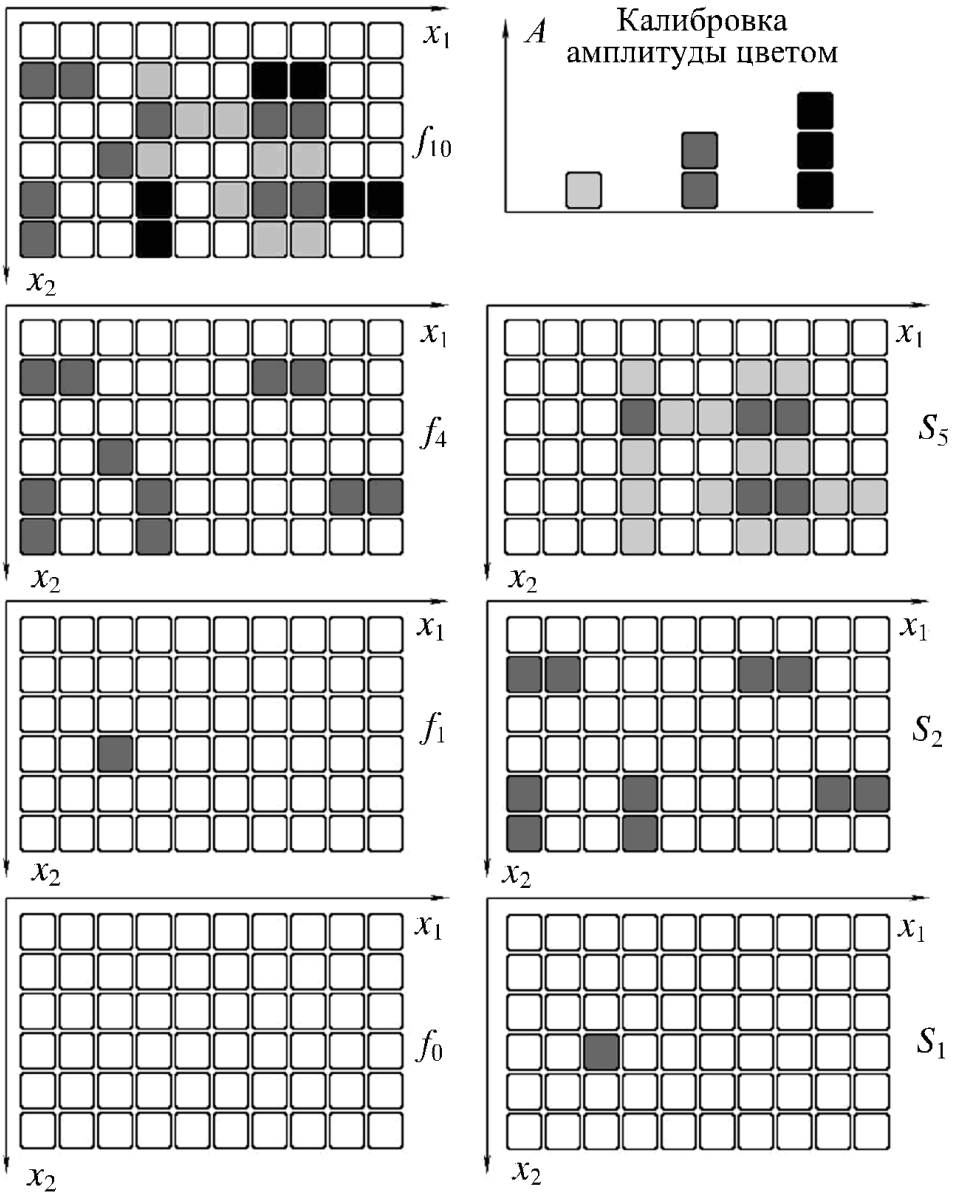


Рис. 5.6. Декомпозиция двумерного сигнала

На рис. 5.7 представлен исходный сигнал (*а*) и спектр двумерного сигнала (*б*, *в*), полученный с помощью функции $f[x, y] = \sin \sqrt{x^2 + y^2}$ на интервале $x, y \in [-15\pi, 15\pi]$. Спектральная плотность p рассчитывалась как отношение суммы всех ненулевых элементов, образующих элементарные всплески, для каждой спектральной составляющей S_i (отдельно по направлениям (x, y)) к площади всей области, в которой задана функция

$f[x, y]$. Очевидно, что максимальное значение p соответствует протяжённости всплесков x=15 и y=15 и периоду радиальных волн 2π.

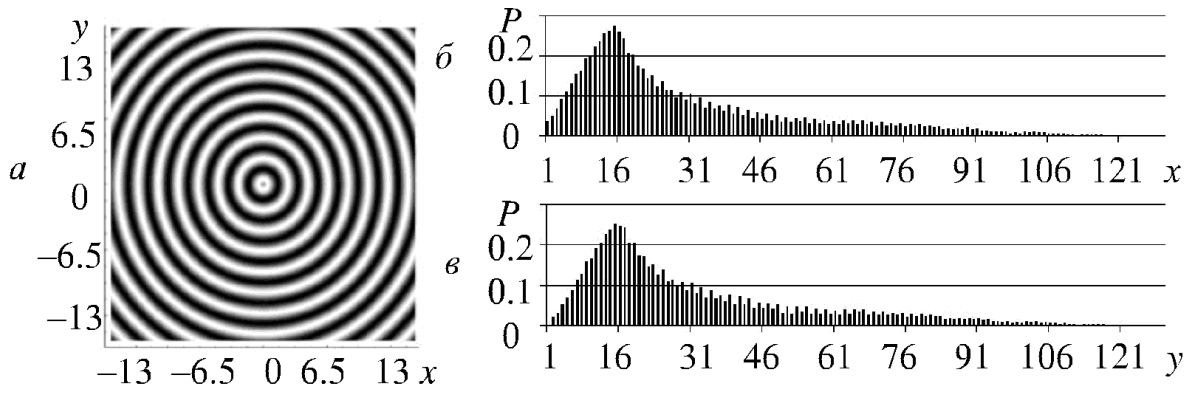


Рис. 5.7. Исходный сигнал и спектр изображения радиальных волн, полученных способом декомпозиции по БПВ

5.4. Свойства и особенности преобразования

На рис. 5.8 представлены некоторые варианты конфигурации элементов декомпозиции на примере одномерного сигнала $f[x]$ с амплитудой A. Приведены четыре варианта сигналов и некоторые возможные способы декомпозиции. Перечёркнутыми стрелками обозначены некорректные варианты декомпозиции (остальными стрелками – корректные).

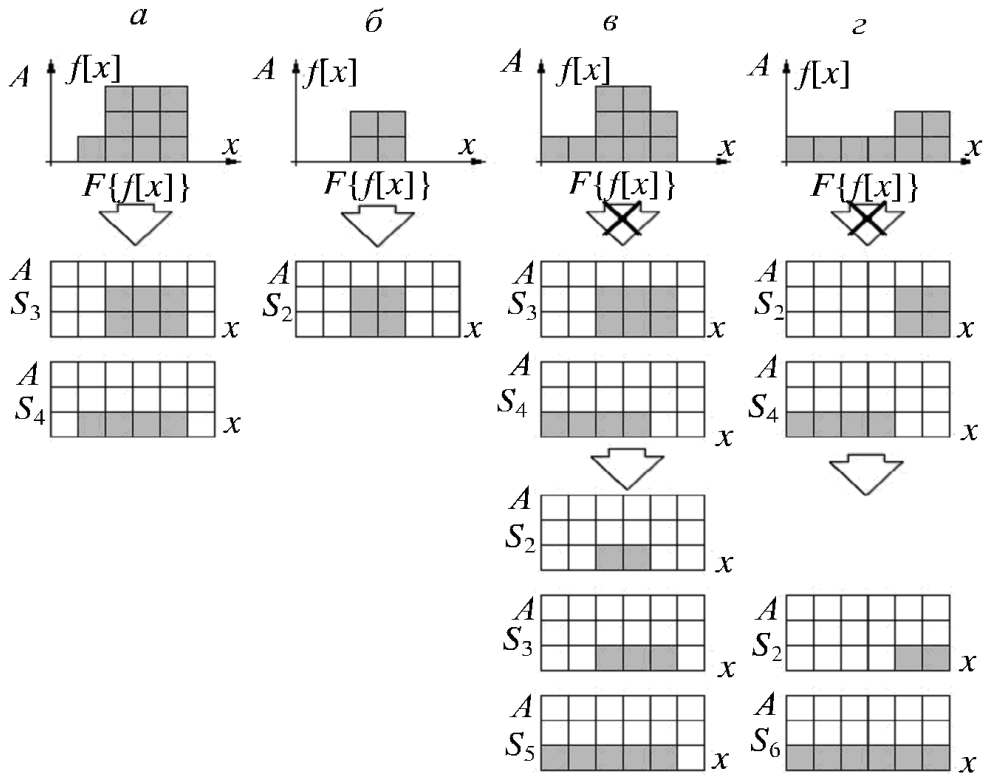


Рис. 5.8. Варианты конфигурации элементов декомпозиции по БПВ

Проанализировав корректные и некорректные варианты декомпозиции [45], можно сделать некоторые обобщения [35]:

– менее протяжённый элементарный всплеск может быть расположен *только полностью* «над» непрерывным, более протяжённым, либо над «нулевым» (отсутствующим) всплеском;

– между всплесками должен быть разрыв, минимальное значение которого равно единице дискретизации сигнала в данном направлении.

Указанные обобщения декомпозиции можно отнести и на случай многомерного сигнала. Согласно второму обобщению, можно ввести понятие «период элементарного всплеска» $T = k + 1$ – минимально допустимый период повторения элементарных всплесков заданной протяжённости k.

Покажем, что декомпозиция по БПВ является линейным преобразованием, т.е. обладает свойствами линейности: аддитивностью и однородностью. Аддитивность обусловлена тем, что суммирование сигналов $f[x]+g[x]$ в пространственной области эквивалентно суммированию сигналов в пространственно-частотной области $F\ f[x]\ + F\ g[x]$, причём при суммировании сигналов в частотной области необходимо приводить результат суммирования к конечному виду в соответствии с обобщениями. Аддитивность преобразования является следствием равенства $f[x]+g[x]=F^{-1}\ F\ f[x]\ + F\ g[x]$ для каждого конкретного значения x и одновременно инвариантности декомпозиции по БПВ.

Однородность декомпозиции по БПВ $F\ mf[x]\ = mF\ f[x]$ является следствием: $m\sum_{k=1}^{K} S_k[x] = mf[x]$, где m – рациональное число.

Для *n*-мерного пространства можно записать свойства линейности преобразования по БПВ:

– аддитивность:

$$f[R^n]+g[R^n]=F^{-1}\ F\ f[R^n]\ + F\ g[R^n]\ ; \qquad (5.3)$$

– однородность:

$$m\sum_{k=1}^{K} S_k[R^n] = mf[R^n]. \qquad (5.4)$$

Стоит отметить, что декомпозиция по БПВ не является инвариантом относительно сдвига сигнала $f[R^n]$, поскольку сдвиг исходного сигнала вызовет соответствующее смещение положения элементарных всплесков.

Если сигнал расположен в некотором закольцованном пространстве, т.е. за максимальным значением координаты направления следует минимальная, и таким же образом осуществляется сдвиг исходного сигнала

$f[R^n]$, то инвариантом является спектральная плотность p, рассчитываемая как отношение суммы всех элементов каждой спектральной составляющей $S_k[R^n]$ к числу всех дискретных элементов, в которой задана функция $f[R^n]$.

Покажем, что при преобразовании по БПВ информация об энергии исходного сигнала сохраняется, что в определенном смысле эквивалентно равенству Парсеваля, энергия исходного сигнала может быть определена как:

$$E = \sum_X f^2[x], \tag{5.5}$$

из выражения (5.2) следует, что:

$$f^2[x] = \left(\sum_{k=1}^{K} S_k[x] \right)^2. \tag{5.6}$$

Раскрывая выражение (5.6) как полиномиальный многочлен второй степени, получаем:

$$\left(\sum_{k=1}^{K} S_k[x] \right)^2 = \sum_{k=1}^{K} S_k^2[x] + 2 \sum_{i=1}^{K-1} \sum_{j=i+1}^{K} S_i[x] S_j[x], \tag{5.7}$$

в итоге можно записать:

$$\sum_X f^2[x] = \sum_X \left[\sum_{k=1}^{K} S_k^2[x] + 2 \sum_{i=1}^{K-1} \sum_{j=i+1}^{K} S_i[x] S_j[x] \right]. \tag{5.8}$$

Правую часть, стоящую под общей суммой выражения (5.8), можно представить в виде:

$$\sum_{k=1}^{K} S_k^2[x] + 2 \sum_{i=1}^{K-1} \sum_{j=i+1}^{K} S_i[x] S_j[x] = \sum_{i=1}^{K} \sum_{j=1}^{K} S_i[x] S_j[x]. \tag{5.9}$$

Далее, в соответствии с выражением (5.9) можно отобразить «таблицу произведения»: на рис. 5.9 приведён пример для случая $S_k \neq 0$. Заметим, что произведения спектральных составляющих $S_i \times S_j = S_j \times S_i$ имеют протяжённость наименьшего всплеска $\min(i, j)$ и размерность квадрата амплитуды (A^2). После извлечения корня из суммы произведений $S_i \times S_j = S_j \times S_i$, выделенных по признаку равных протяжённостей, мы получим взаимно-ортогональные элементы спектральных составляющих:

$$\overline{S}_i[x] = \sqrt{S_i^2[x] + 2 \sum_{j=i+1}^{K} S_i[x] S_j[x]}, \tag{5.10}$$

причём в силу ортогональности (выполняется равенство Парсеваля как обобщение теоремы Пифагора для *n*-мерного случая) обратное преобразование будет иметь вид:

$$f[x] = \sqrt{\sum_{i=k}^{K} \overline{S}_i^2[x]} \,. \tag{5.11}$$

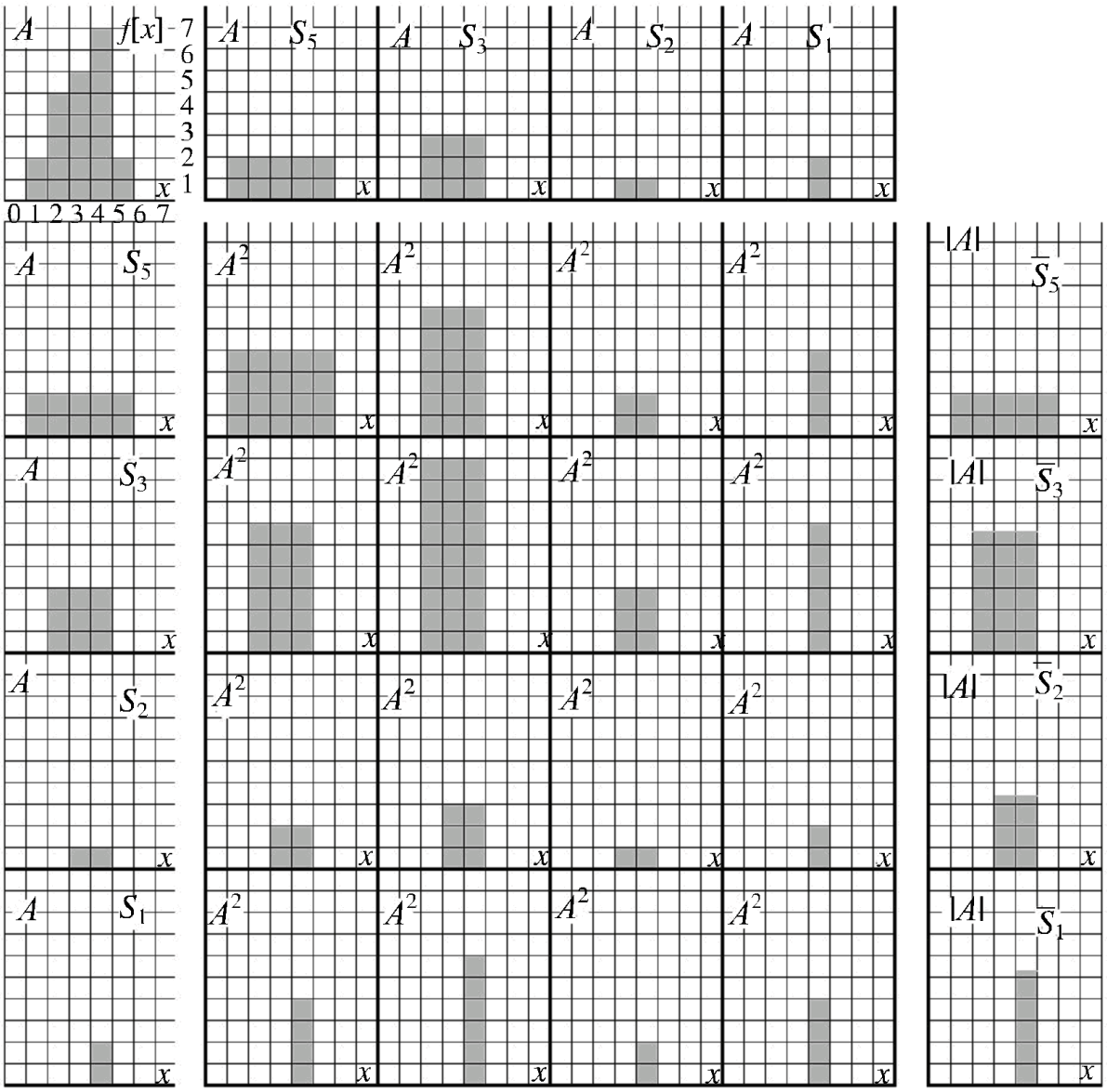


Рис. 5.9. Получение ортогональной формы декомпозиции по БПВ

Преобразуем выражение (5.8) для $\overline{S}_i[x]$:

$$\sum_X f^2[x] = \sum_X \left[\sum_{k=1}^{K} \overline{S}_k^2[x] \right], \tag{5.12}$$

для пространства R^n:

$$\sum_{R^n} f^2[R^n] = \sum_{R^n}\left[\sum_{k=1}^{K} \bar{S}_k^2[R^n]\right]. \qquad (5.13)$$

Рассмотрим в качестве примера переход к ортогональной форме $\bar{S}_i[x]$ для сигнала, представленного на рис. 5.9. Вначале рассчитывается множество элементов спектральных составляющих $S = \langle S_5, S_3, S_2, S_1 \rangle$. Далее формируются элементы произведений $S \times S$ (упорядоченные пары), для которых выполняется свойство коммутативности $S_i \times S_j = S_j \times S_i$. Результат произведений отображён в центральной части рис. 5.9. В качестве примера рассчитаем значения амплитуд спектральных элементов декомпозиции для 4-го отсчёта (производится слева, начиная с нуля по оси x, шкала отображена на графике $f[x]$) $\bar{S}_5[4] = \sqrt{2^2} = 2$, $\bar{S}_3[4] = \sqrt{3^2 + 2 \cdot 3 \cdot 2} = \sqrt{21}$, $\bar{S}_2[4] = \sqrt{1^2 + 2 \cdot 1 \cdot 2 + 2 \cdot 1 \cdot 3} = \sqrt{11}$, $\bar{S}_1[4] = \sqrt{2^2 + 2 \cdot 2 \cdot 2 + 2 \cdot 2 \cdot 3 + 2 \cdot 2 \cdot 1} = 2\sqrt{7}$. Результаты расчётов $\bar{S}_i[x]$ приведены в правом столбце (рис. 5.9). Очевидно, что амплитуды исходного сигнала по значениям $\bar{S}_i[x]$ могут быть восстановлены в соответствии с выражением (5.11). Далее рассчитаем полную энергию E в соответствии с (5.12):

$$\sum_{X}\left[\sum_{k=1}^{K} \bar{S}_k^2[x]\right] = 5 \cdot 4 + 3 \cdot 21 + 2 \cdot 11 + 1 \cdot 28 = 133,$$

вычисление по выражению (5.5) даёт:

$$\sum_{X} f^2[x] = 0^2 + 2^2 + 5^2 + 6^2 + 8^2 + 2^2 + 0^2 + 0^2 = 133.$$

Вычислительную сложность алгоритма прямого преобразования по БПВ для одномерного случая можно оценить как

$$\mathbf{O}\ N\ , \qquad (5.14)$$

где N – размер массива данных.

Для прямого преобразования по БПВ сигнала $f[R^n]$, ограниченного размерами $X_1, X_2, ..., X_n$ пространства $x_1, x_2, \ldots, x_n$, в котором задан сигнал, вычислительную сложность можно оценить как:

$$\mathbf{O}\ nX_1X_2\times,...,\times X_n\ . \qquad (5.15)$$

Отметим, что преобразования в соответствии с (5.1) и (5.2) могут быть выполнены на множестве целых чисел, что обеспечивает высокое быстродействие и простоту реализации преобразования по БПВ полностью аппаратными средствами.

Области применения способа декомпозиции по БПВ определяют ориентированность на цифровые сигналы, возможность декомпозиции мно-

гомерных сигналов, получение пространственно определённого спектра сигнала $f[R^n]$, т.е. отображение сигнала в фазовое пространство. Рассмотренный способ преобразования по БПВ был успешно применён в задачах распознавания образов и анализе частотного спектра графических изображений.

6. Предпосылки к формированию компактного вида цифровых сигналов

В современном мире очень остро стоит проблема загруженности каналов связи при значительном времени простоя вычислительных мощностей. Ценность информации заключается, в частности, в ее доступности, что требует оперативной передачи, а по возможности и хранения на локальных ресурсах, потенциал которых в большинстве случаев сильно ограничен. Для поддержания актуальности данных необходимо синхронизировать источники информации. Такие факторы требуют разработки оптимальных способов форматирования, а следовательно, передачи информации – не только в пространстве, но и во времени (хранение). Потому возникает задача уменьшения объёма – *сжатия* (*компрессии*) [105, 107] – передаваемых данных с сохранением информации.

6.1. Общие принципы и классификация

Эффективность применения методов и алгоритмов сжатия характеризуется *коэффициентом сжатия* (измеряется в единицах bpb: bit per bit – бит на бит):

$$k = \frac{D_{\text{out}}}{D_{\text{in}}}, \tag{6.1}$$

где D_{out} и D_{in} – объём сжатых (выходных) и исходных (входных) данных ($k < 1$ означает сжатие информации). Величина, обратная коэффициенту сжатия, – *фактор сжатия*:

$$c = \frac{1}{k} = \frac{D_{\text{in}}}{D_{\text{out}}}. \tag{6.2}$$

Современные методы сжатия информации требуют мощного математического аппарата для анализа специфических особенностей информации с целью повышения её энтропии (т.е. плотности). Методы компактного представления медиаинформации (изображения, звуковые и видеоданные) базируются на устранении ее избыточности, в том числе исходя из психофизиологических особенностей восприятия.

Под *избытком информации* понимают ту часть объёма данных, которая может быть исключена специальными методами кодирования при восстановлении из сжатого вида (декодировании).

Стоит отметить, что при снижении избыточности возрастает вероятность повреждения информации, однако сжатая информация может содержать дополнительные данные для восстановления в случае повреждений. Таким образом, можно выделить два принципиально различных класса методов сжатия информации:

– *без потерь* – применяется в тех случаях, когда любое искажение информации при декодировании недопустимо, пример – управляющие кодовые последовательности (исполняемые коды), текстовые данные, некоторые виды изображений, результаты некоторых исследований, наблюдений. Сжатие без потерь обеспечивают алгоритмы Шеннона–Фано, Зива–Лемпела (LZ78), Хаффмана, арифметического кодирования, словарного кодирования и др.;

– *с потерями* – применяется в случаях, когда частичная потеря информации допустима [17], например, фотографии, звуковые и видеоданные. Соответствующие алгоритмы обычно строятся на анализе информации для выявления пространственных и (или) частотных характеристик сигнала с последовательным исключением некоторой части информации.

Для оценки различий восстановленного после сжатия с потерями информации $f[x]$ и исходного $r[x]$ сигналов используют несколько критериев:

– среднеквадратическая ошибка (Mean Square Error, MSE):

$$\mathrm{MSE} = \frac{1}{n}\sum_{i=1}^{n}(f[x]-r[x])^2 \; ; \tag{6.3}$$

– пиковое отношение сигнал/шум (Peak Signal to Noise Ratio, PSNR):

$$\mathrm{PSNR} = 20\log_{10}\frac{\max_i(|f[x]|)}{\sqrt{\mathrm{MSE}}} \; . \tag{6.4}$$

– отношение сигнал/шум (Signal to Noise Ratio, SNR):

$$\mathrm{SNR} = 20\log_{10}\frac{\sqrt{\frac{1}{n}\sum_{i=1}^{n}f[x]^2}}{\mathrm{RMSE}} \; , \tag{6.5}$$

где $\mathrm{RMSE} = \sqrt{\mathrm{MSE}}$.

Существуют и другие критерии ошибок. Отметим, что в каждом конкретном случае вопрос о допустимости частичной потери информации и её величине может решаться отдельно.

6.2. Частотно-пространственные области концентрации информации

Различная природа цифровых сигналов определяет их особенности. Принято выделять сигналы, основной объем информации в которых сосредоточен либо в частотной, либо в пространственной области. Информация звукового сигнала преимущественно сконцентрирована в частотной области, графического изображения – в пространственной. При таком подходе необходимо учитывать специфику конкретного сигнала или некоторой группы сигналов. Существуют сигналы, основная информация которых сосредоточена в пространстве, например, дельта-функция при чёткой

пространственной локализации имеет неограниченный частотный спектр. В частотном диапазоне сконцентрирована превалирующая часть информации некоторых графических изображений, к примеру, графиков волновых поверхностей. Говоря о преимущественной области концентрации информации, часто подразумевают специфику восприятия информации человеком или техническим устройством. Например, на восприятие человеком звука больше влияет частота, чем фаза, для технических средств – наоборот, например, эхолот по разности фаз (запаздыванию) может определять расстояние, а по сдвигу частоты, на основании эффекта Допплера – скорость объекта отражения.

Частотные характеристики сигнала могут быть получены с помощью декомпозиции (Фурье, вейвлет и др.), пространственные – с помощью геометрической интерпретации сигнала, определения его производных, вейвлет-анализа, оконного преобразования Фурье.

Используя особенности концентрации информации, можно исключить некоторую «избыточную» часть из частотного или пространственного описания сигнала, что приведет к допустимым искажениям.

Методы получения частотных и пространственных характеристик и исключения некоторой «избыточной» части позволяют синтезировать компактную форму информационной составляющей сигнала. Многие способы перевода информации в компактную форму не обладают универсальностью: конкретный метод получения компактной формы, применимый к одному сигналу или группе сигналов, может быть малоэффективен для других сигналов. Неслучайно в современном мире цифровой обработки сигналов столь много разнообразных форматов компактного хранения данных, это обусловлено не только сражением за коммерческие интересы корпораций на рынке путём ограничения прав на использование с помощью механизмов патентного права, монополизации производства программного и аппаратного обеспечения, но и особенностями самих сигналов.

Значительный интерес для разработки способов компактного хранения представляют фотографии, видеофильмы, сигналы цифрового телевидения и прочие многомерные сигналы, сопряжённые с передачей графической информации. В большинстве случаев графические сигналы занимают существенные ресурсы памяти при значительном количестве избыточной информации. Применение только статистических способов сжатия (таких как алгоритм Хаффмана, арифметическое кодирование) или сжатия серий в большинстве случаев не даёт существенного эффекта. Поэтому применяют различные совокупности способов сжатия, часть которых основана на преобразовании передаваемой сигналом информации к компактной форме путем анализа сигнала как поля (спектральный анализ, анализ геометрической структуры, корреляционный анализ, дифференциальный анализ и др.).

6.3. Предпосылки к исключению избыточной информации на базе частотно-дифференциального анализа

Рассмотрение цифровых сигналов как полевых структур позволяет использовать численные методы расчёта полей в их анализе и преобразовании. Основы методов частотного анализа цифровых сигналов, в том числе с целью последующего сжатия, рассмотрены в классической литературе по ЦОС [85, 95, 112, 114], поэтому обратим внимание на пространственный анализ сигналов. Существует ряд способов формирования компактного вида цифровых сигналов с помощью анализа пространственной структуры. В частности, для получения компактной формы изображений применяют векторизацию, т.е. в ходе морфологического анализа изображения (в большинстве случаев это достаточно сложная вычислительная задача) выявляют графические примитивы (прямые, окружности, прямоугольники и т.д.). Наиболее эффективно таким способом можно сжимать техническую графику (чертежи, эскизы и др.) и некоторые виды изображений, объекты, которых имеют чёткие границы и относительно простые формы. Следующим способом является фрактальное сжатие, основанное на том, что изображение можно представить более компактно с помощью коэффициентов системы итерируемых функций (фракталов) [171]. Фрактальное сжатие очень затратно с точки зрения вычислений, но на некоторых образцах изображений показывает очень хорошие результаты. Известен способ формирования из изображения с помощью метода «brush fire» (лесной пожар) скелета многоугольника толщиной 1–2 пиксела – диаграммы Вороного [110]. Этот способ также достаточно затратен и не получил широкого распространения. Существуют и другие способы пространственного анализа цифровых сигналов с целью получения компактной формы.

Рассмотрим способ, основанный на анализе дифференциальной структуры цифрового сигнала. Под дифференциальной структурой подразумеваются производные, обычно до третьего порядка, цифрового сигнала, для многомерных сигналов – частные производные. В самом простом случае дифференциальное кодирование строится следующим образом: для элементов цифрового сигнала $f[x]$, $x \in X$, вычисляется ряд дельта-кода $d[x] = f[x+1] - f[x]$, $x \in (X-1)$. В силу того что разность значений соседних элементов исходного сигнала $f[x]$ обычно существенно меньше диапазона его значений, полученный ряд может быть эффективно сжат [105]. Для восстановления исходного сигнала необходимо кроме ряда дельта-кода знать начальные условия. На рис. 6.1 получена диаграмма дельта-кода, в табл. 6.1 – ряды численных значений. Очевидно, что значения дельта-кода имеют меньшую амплитуду и для их кодирования потребуется меньшее

число разрядов, кроме того, они чаще повторяются, следовательно, к ним может быть применено статистическое кодирование.

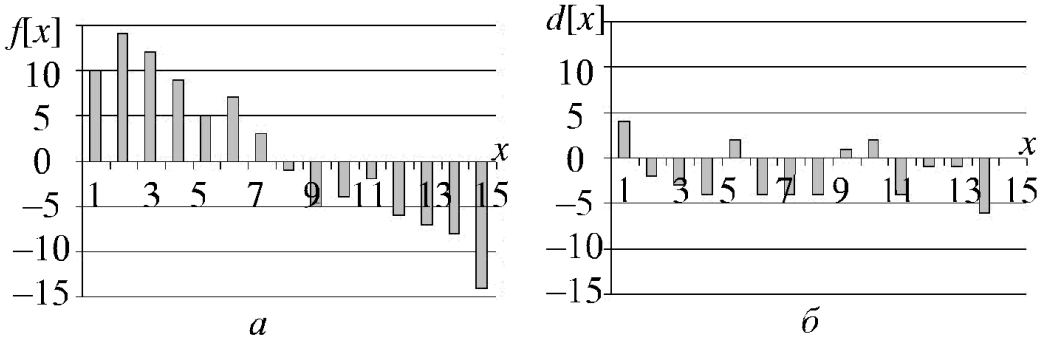


Рис. 6.1. а – исходный сигнал, *б* – дельта-код

Но далеко не все типы сигналов могут быть успешно сжаты с помощью дельта-кода, дело в том, что эффективно могут сжиматься достаточно гладкие функции, к примеру, полученные в результате оцифровки реальных физических сигналов: звук, фото- и видеоизображения и т.д.

Таблица 6.1. Значения функции $f[x]$ и её дельта-кода $d[x]$

x	$f[x]$	$d[x]$
1	10	4
2	14	–2
3	12	–3
4	9	–4
5	5	2
6	7	–4
7	3	–4
8	–1	–4
9	–5	1
10	–4	2
11	–2	–4
12	–6	–1
13	–7	–1
14	–8	–6
15	–14	—

Большинство сигналов, полученных в результате регистрации реальных физических процессов, могут быть описаны, с той или иной степенью точности, уравнениями, в частности, дифференциальными.

Рассмотрим основные типы уравнений математической физики с частными производными второго порядка для дву- и трёхмерных пространств (x, y – координаты пространства, t – координата времени) [97, 103].

Волновые уравнения:

$$\frac{\partial^2 f(x,t)}{\partial t^2} = a^2 \frac{\partial^2 f(x,t)}{\partial x^2}, \tag{6.6}$$

$$\frac{\partial^2 f(x,y,t)}{\partial t^2} = a^2 \left(\frac{\partial^2 f(x,y,t)}{\partial x^2} + \frac{\partial^2 f(x,y,t)}{\partial y^2} \right). \tag{6.7}$$

К уравнениям этого типа приводит рассмотрение колебаний газа, электромагнитного поля в проводнике, поперечных колебаний струны, продольных – стержней, крутильных – валов и т.д. Один из вариантов эквивалентного решения (6.6) получен при помощи гармонического анализа в 1824 г. Ж. Фурье. Отметим, что не все процессы, квалифицируемые современной физикой как волновые, описываются приведёнными волновыми уравнениями, примерами может быть распространение волн в нелинейной среде, солитоны и др.

Уравнения теплопроводности, или уравнения Фурье:

$$\frac{\partial f(x,t)}{\partial t} = a^2 \frac{\partial^2 f(x,t)}{\partial x^2}, \tag{6.8}$$

$$\frac{\partial f(x,y,t)}{\partial t} = a^2 \left(\frac{\partial^2 f(x,y,t)}{\partial x^2} + \frac{\partial^2 f(x,y,t)}{\partial y^2} \right). \tag{6.9}$$

К этим уравнениям приводит рассмотрение процессов теплопроводности, фильтрации жидкости и газов в пористой среде, некоторые задачи теории вероятностей и т.д.

Уравнения Лапласа и Пуассона:

$$\frac{\partial^2 f(x,y)}{\partial x^2} + \frac{\partial^2 f(x,y)}{\partial y^2} = \varphi(x,y), \tag{6.10}$$

$$\frac{\partial^2 f(x,y,z)}{\partial x^2} + \frac{\partial^2 f(x,y,z)}{\partial y^2} + \frac{\partial^2 f(x,y,z)}{\partial z^2} = \varphi(x,y,z). \tag{6.11}$$

Уравнения этого типа позволяют выполнять моделирование электромагнитных полей, стационарных и переходных тепловых состояний, задач гидродинамики, диффузии и т.д. Уравнения Пуассона (6.10) и (6.11) при равенстве правой части нулю (для всех значений x, y, z) обращаются в уравнения Лапласа.

Таким образом, большинство процессов (стационарных и динамических), которые являются первичными источниками сигналов, могут быть аналитически описаны дифференциальными уравнениями. Важно заметить,

что дифференциал и производная функции, в соответствии с теоремой о почленном дифференцировании [67], связаны с рядом Фурье.

Теорема. Пусть функция $f(x)$ *и все её производные до некоторого порядка* m (m *– целое неотрицательное число) непрерывны на сегменте* $[-\pi,\pi]$ *и удовлетворяют условиям:*

$$\begin{cases} f(-\pi)=f(\pi), \\ f'(-\pi)=f'(\pi), \\ \ldots \\ f^{(m)}(-\pi)=f^{(m)}(\pi), \end{cases}$$

тогда тригонометрический ряд Фурье функции $f(x)$ *можно* m *раз почленно дифференцировать на сегменте* $[-\pi,\pi]$.

Можно записать выражение, связывающее производную порядка p (p=1, 2,…, m) и члены ряда Фурье функции $f(x)$ [67]:

$$f^{(p)}(x)=\sum_{k=1}^{\infty}k^{p}\left[A_k\sin\left(kx-\frac{\pi p}{2}\right)+B_k\cos\left(kx-\frac{\pi p}{2}\right)\right], \qquad (6.12)$$

где A_k, B_k – коэффициенты ряда Фурье функции $f(x)$ (4.25). Из уравнения (6.12) следует, что производные более высокого порядка больше зависят от высокочастотных составляющих ряда Фурье.

Особая значимость теоремы (6.12) состоит во взаимозависимости гармонического ряда отображения функции и её производной, это свойство обеспечивает предпосылки для композитного сжатия цифровых сигналов, когда один метод позволит производить сжатие в пространстве-времени и частоте одновременно. Вейвлет-анализ, как и оконное преобразование Фурье, интересен тем, что позволяет получить не только частотную характеристику, но пространственную локализацию спектра.

6.4. Сжатие битовых последовательностей без потерь

Обычно, говоря о *сжатии битовых последовательностей*, имеют в виду способы сжатия без потерь. В общем случае, поскольку цифровой сигнал можно рассматривать как битовую последовательность, любой способ сжатия, в том числе и с потерями, можно соотносить со способом сжатия битовых последовательностей.

Особенно эффективно происходит сжатие битовых полей, содержащих значительное число повторяющихся элементов (0 или 1) или серий (упорядоченных множеств элементов – кортежей). Сжатие битовых последовательностей может быть эффективно применено для битовых матриц

изображений, звуковых данных, текстовых сообщений, управляющих кодов.

Можно выделить два класса способов сжатия битовых последовательностей: с учётом и без учёта статистических особенностей. Для способов первого класса, к которым относятся коды Шеннона–Фано, код Хафмена, арифметическое кодирование и др., требуется предварительный анализ частоты повторения некоторых серий с целью формирования кодовых таблиц. Недостаток этих способов заключается в необходимости статистического анализа данных и передачи кодовой таблицы вместе с данными. Ко второму классу способов сжатия битовых последовательностей относят способы, не учитывающие статистические закономерности распределения элементов данных в сигнале.

В качестве примера рассмотрим сжатие битовых серий на основе системы счисления Фибоначчи[14].

Напомним, что *префиксным* называется код, ни одно слово которого не может быть начальной строкой другого слова кода. *Суффиксным кодом* называют такой, ни одно слово которого не может быть конечной строкой другого слова кода [6].

Ряд Фибоначчи образован рекуррентной последовательностью вида:

$$\begin{cases} F_i = F_{i-1} + F_{i-2}, \\ F_0 = 1, \\ F_1 = 1. \end{cases} \qquad (6.13)$$

Система счисления на базе чисел Фибоначчи построена следующим образом: любое неотрицательное число представлено единственным образом при помощи упорядоченной последовательности единиц и нулей, каждый элемент такой бинарной последовательности соответствует числу из ряда Фибоначчи, причём число может быть преобразовано в десятичную систему счисления при помощи выражения:

$$b_{10} = \sum_{1}^{N} a_i F_i \ , \qquad (6.14)$$

где a_i – элемент последовательности в соответствующем разряде записи числа. Ещё одной важной особенностью, исключающей несколько вариантов записи одного численного значения, является недопустимость единиц в соседних разрядах, т.е. в соответствии с рекуррентным выражением (6.13) при наличии двух идущих подряд единиц происходит перенос единицы в старший разряд. Например, для числа, записанного при помощи

[14] Леонардо Пизанский (≈1170–1250, Пиза) – первый крупный математик средневековой Европы, наиболее известен под именем Фибоначчи.

системы счисления Фибоначчи вида 10011001 (младший разряд справа), правильной будет запись 10100001.

Таблица 6.2. Соответствие записи чисел от 0 до 7 в различных системах счисления

Число в десятичной системе счисления	Число в двоичной системе счисления	Образующие числа Фибоначчи	Число в системе счисления Фибоначчи
0	0	–	0
1	1	$F(1)$	1
2	10	$F(2)$	10
3	11	$F(3)$	100
4	100	$F(3)+F(1)$	101
5	101	$F(4)$	1000
6	110	$F(4)+F(1)$	1001
7	111	$F(4)+F(2)$	1010
8	1000	$F(5)$	10000
9	1001	$F(5)+F(1)$	10001
10	1010	$F(5)+F(2)$	10010
11	1011	$F(5)+F(3)$	10100
12	1100	$F(5)+F(3)+F(1)$	10101
13	1101	$F(6)$	100000
14	1110	$F(6)+F(1)$	100001
15	1111	$F(6)+F(2)$	100010

Следует отметить, что существует несколько разновидностей систем счисления, построенных на ряде Фибоначчи, в которых возможны идущие подряд единицы.

Отсутствие в записи числа идущих подряд единиц (11) позволяет создать систему кодирования – префиксную или суффиксную – в зависимости от положения младшего разряда (справа или слева) и выбрать начало последовательности данных сообщения. Запись числа в системе счисления Фибоначчи в старшем разряде (при отсечении незначащих разрядов) всегда будет иметь единицу (кроме записи нулевого значения), младший разряд может содержать единицу или нуль. Таким образом, для образования (префиксной или суффиксной) последовательности конкатенацию единицы необходимо производить к старшему разряду. Такая система кодирования может быть применена для сжатия битовых серий [145]. Рассмотрим пример. Пусть имеется битовая последовательность

011000111100000111111000000011111111 (36 символов),

соответствующий ей префиксный код, согласно табл. 6.2, будет иметь вид:

11110110011011100011001110101100000 (34 символа).

Поскольку в исходном сообщении присутствуют последовательности единиц и нулей небольшой протяжённости, сжатие незначительно.

Длину кода на базе системы счисления Фибоначчи можно оценить в соответствии с выражением:

$$l = \left\lfloor \log_{\frac{1}{2}(1+\sqrt{5})} \sqrt{5}N + 1 \right\rfloor, \tag{6.15}$$

где N – число повторяющихся битов. Сжатие будет тем более эффективным, чем больше протяжённость повторяющихся серий битов последовательностей. Средняя длина кода для последовательности серий протяжённостью N_1, N_2, ..., N_p:

$$\overline{l} = \frac{1}{p}\sum_{i=1}^{p} \left\lfloor \log_{\frac{1}{2}(1+\sqrt{5})} \sqrt{5}N_i + 1 \right\rfloor. \tag{6.16}$$

Условие эффективности сжатия можно записать следующим образом:

$$\sum_{i=1}^{p} \left(\left\lfloor \log_{\frac{1}{2}(1+\sqrt{5})} \sqrt{5}N_i + 1 \right\rfloor - N_i \right) < 0. \tag{6.17}$$

В соответствии с выражением (6.1) получим коэффициент сжатия

$$k = \frac{\sum_{i=1}^{p} \left\lfloor \log_{\frac{1}{2}(1+\sqrt{5})} \sqrt{5}N_i + 1 \right\rfloor}{\sum_{i=1}^{p} N_i}. \tag{6.18}$$

7. Сжатие и синтез многомерных сигналов с помощью анализа дифференциальной структуры

В пространственной и частотной областях представления некоторого сигнала плотность информации может существенно различаться. Многие методы сжатия с потерями исключают часть информации из частотной, но не из пространственно-временной области. Из разнообразных цифровых сигналов, в частности изображений, можно выделить такие, для которых исключение информации из пространственной области обеспечивает результат не хуже, а в некоторых случаях лучше, чем исключение информации из частотной области. Специфика предлагаемого метода состоит в исключении некоторой информации из пространственной или пространственно-временной области. Многопоточность современных вычислительных средств позволяет реализовать рассматриваемый метод с приемлемым быстродействием.

7.1. Анализ дифференциальной структуры и формирование паттерна краевых условий цифрового сигнала

Введем определения некоторых понятий. Совокупность условий, описывающих состояние системы в пространстве на границах и в некоторых внутренних областях, называют *граничными условиями*; совокупность, описывающая состояние системы в некоторые моменты времени (обычно начальные $t = 0$), называется *начальными условиями*. Совокупность граничных и начальных условий называется *краевыми условиями*.

Пусть в области $\mathbf{W}$ задан сигнал $f(R^n)$, предположим, что в некоторой части заданной области $\mathbf{U} \subset \mathbf{W}$ $f(R^n)$ удовлетворяет дифференциальному уравнению вида:

$$Af(R^n)=\chi(R^n), \qquad (7.1)$$

где A – дифференциальный оператор, $\chi(R^n)$ – функция правой части. Краевые условия $p(R^n)$ заданы в области $\mathbf{B} \subset \mathbf{W}$ таким образом, что $\mathbf{U} \cup \mathbf{B} = \mathbf{W}$. Функцию $p(R^n)$ будем называть *паттерном* исходного сигнала $f(R^n)$. Используя уравнение (7.1) и паттерн сигнала $p(R^n)$, можно с заданной точностью восстановить $f(R^n)$ в области $\mathbf{U}$. Таким образом, анализ дифференциальной структуры сигнала $f(R^n)$ сводится к выбору анализирующего уравнения (или системы уравнений) (7.1) и отыскании паттерна $p(R^n)$.

Отметим, что определение краевых условий путём анализа дифференциальной структуры является обратной задачей численных методов.

Рассмотрим формирование паттерна на примере одномерного сигнала $f[x]$. Положим, что $f(x)$ есть непрерывный прообраз дискретного образа некоторого сигнала $f[x]$, при этом он описывается во всех частях области **U** уравнением:

$$\frac{d^2 f(x)}{dx^2} = 0 . \tag{7.2}$$

Переходя к дискретному образу $f[x_i]$ и полагая, что он задан на сетке $x_0, x_1, ..., x_{n-1} \in \mathbf{W}$ с равномерным шагом $h = x_{i+1} - x_i$, где $i \in 0, ..., n-1$, для всех $x_i \in \mathbf{U}$ можно записать уравнение (7.2) в виде:

$$\frac{d^2 f(x)}{dx^2} \approx \frac{\frac{f[x_{i+1}] - f[x_i]}{h} - \frac{f[x_i] - f[x_{i-1}]}{h}}{h} =$$

$$= \frac{f[x_{i+1}] - 2f[x_i] + f[x_{i-1}]}{h^2} = 0, \tag{7.3}$$

откуда

$$f[x_i] = \frac{f[x_{i+1}] + f[x_{i-1}]}{2} . \tag{7.4}$$

Вычислять паттерн $p[x_i]$ будем по следующей схеме.

1. Положим, что во всей области **W** все значения паттерна равны значениям сигнала: $p[x_i] = f[x_i]$.

2. Выберем некоторое числовое значение Δ-*критерия*, согласно которому будем исключать элементы из паттерна.

3. С учётом (7.4) для всех $f[x_i]$ в области **W** вычислим значения:

$$s = \left| f[x_i] - \frac{f[x_{i+1}] + f[x_{i-1}]}{2} \right| ,$$

если условие $s < \Delta$ истинно, то значение $p[x_i]$ (для соответствующего x_i) исключается из паттерна, в противном случае элемент остаётся в паттерне.

В результате выполнения указанной последовательности действий получен паттерн сигнала $f[x_i]$. Пример формирования паттерна приведен в листинге 7.1, на рис. 7.1 представлены результаты выполнения программы.

Листинг 7.1. Формирование паттерна (MATLAB)

```
%================================================================
clear all;          % удаление переменных
close all;          % закрытие всех окон
clc;                % очистка командного окна
% задание исходного сигнала
data=[1 2 3 4 5 6 4 8 6 4 2 0 -1 -2 -3 -4 -5 ...
      -6 -7 -4 -1 2 3 4 6 7 8 9 7 5 3 2 1 0 -1];
patt=data;          % значения паттерна равны значениям сигнала
del=0.1;            % Δ-критерий для исключения значений из паттерна
% цикл исключения некоторых значений из паттерна
for i=2:1:(length(data)-1)
  s=0.5*(data(i+1)+data(i-1));
  if(abs(s-data(i)) < del)
     % если условие истинно
     % исключение из паттерна
     patt(i)=0;
  elseif(patt(i)==0)
     % исключение неоднозначности
     patt(i)=del;
  end
end
% графики результатов расчётов
figure;
subplot(2,1,1); stairs(data);
title('Сигнал (data)')
subplot(2,1,2); stairs(patt);
title('Паттерн (patt)')
%================================================================
```

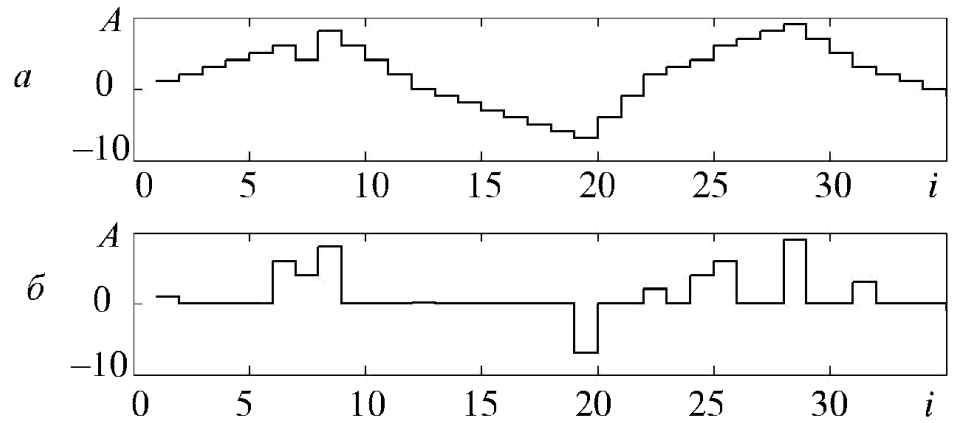


Рис. 7.1. Результаты выполнения программы (листинг 7.1).
а – исходный сигнал, *б* – паттерн

Очевидно, что паттерн (рис. 7.1, *б*) содержит значительно больше равнозначных элементов, чем исходный сигнал, это позволяет его эффективнее сжать. Приравненное нулю значение элемента паттерна служит признаком его исключения (`patt(i)=0;`), такой принцип может привести к неоднозначности, если, например, в исходном сигнале присутствуют нулевые

значения, – ведь для восстановления сигнала необходимо точно знать, принадлежит конкретный элемент паттерна краевым условиям или нет. Избежать неоднозначности трактовки паттерна поможет изменение значений отсчетов в исходном сигнале. Допустимый форматом данных выход значений сигнала из определенного диапазона – признак того, что элемент не является частью краевых условий. В программе (см. листинг 7.1) неоднозначность исключается путём изменения нулевых значений сигнала (нулей), вошедших в паттерн, на величину Δ -критерия (`del`).

Изменяя значение (`del`), можно управлять числом элементов, вошедших в паттерн, от этого изменения будет зависеть количество исключаемой информации и степень сжатия. Для точного восстановления сигнала требуется принять значение (`del`) меньше шага квантования сигнала и в некоторых случаях сохранять, например, в виде битовой маски данные, позволяющие восстановить нулевые (или другие «спорные») значения в сигнале. На рис. 7.2 приведены исходный сигнал и его поле паттернов, полученное в результате объединения серии паттернов (для различных значений Δ -критерия) в массив. Чёрный цвет соответствует нулевым элементам паттерна, не являющимся краевыми условиями, оттенки серого – краевым условиям. При увеличении значения (`del`) происходит уменьшение числа элементов паттерна, принадлежащих множеству краевых условий.

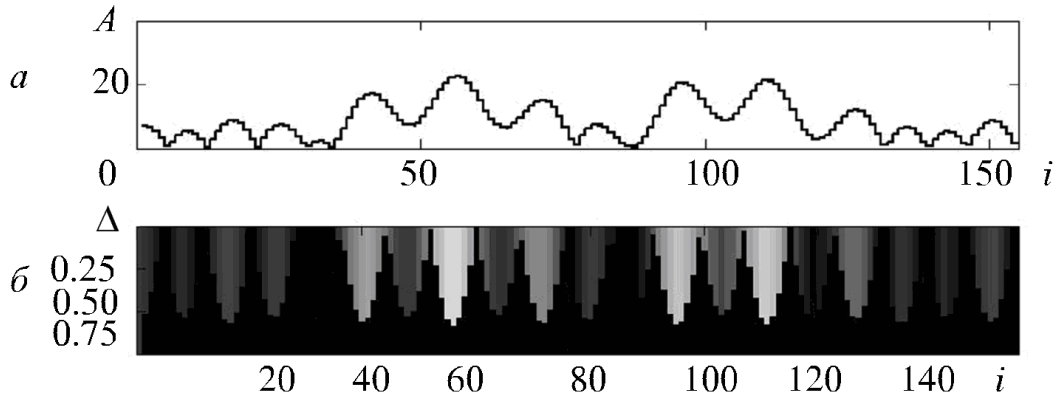


Рис. 7.2. Зависимость поля паттернов сигнала от Δ-критерия.
а – исходный сигнал, *б* – поле паттернов

Паттерн, содержащий относительно небольшое число элементов, являющихся краевыми условиями, позволяющими вполне точно восстановить исходный сигнал, может быть получен только для гладких функций. Следовательно, применение данного способа с целью сжатия шумоподобных сигналов неэффективно.

Изменяя порядок производных дифференциального оператора A (7.1) и выбирая некоторые функции $u(R^n)$, можно влиять на качество

восстановления сигнала и число исключённых из него элементов, т.е. размер паттерна, что, в свою очередь, скажется на эффективности сжатия.

7.2. Восстановление сигнала методом конечных разностей

Восстановление исходного сигнала осуществляется *методом конечных разностей* (МКР) при помощи соотношения, выбранного в качестве анализирующего дифференциального уравнения, и краевых условий, содержащихся в паттерне. Программа (листинг 7.2) и результаты её выполнения (рис. 7.3) иллюстрируют возможность анализа дифференциальной структуры с формированием паттерна и последующего восстановления сигнала.

Листинг 7.2. Формирование паттерна и восстановление сигнала (MATLAB)

```
%=================================================================
clear all;        % удаление переменных
close all;        % закрытие всех окон
clc;              % очистка командного окна
% задание исходного сигнала
x = -pi:0.04:pi;
data = abs(10*sin(x)+8*sin(2*x)+7*cos(12*x));
% формирование паттерна
patt = data;     % значения паттерна равны значениям сигнала
del = 0.7;       % Δ-критерий для исключения значений из паттерна
% цикл исключения некоторых значений из паттерна
for i=2:1:(length(data)-1)
  s=0.5*(data(i+1)+data(i-1));
  if(abs(s-data(i)) < del)
     % если условие истинно, исключение из паттерна
     patt(i)=0;
  elseif(patt(i)==0)
     patt(i)=del;
  end
end
% восстановление сигнала по паттерну методом конечных разностей
result = patt;
for j=0:1:10
  for i=2:1:(length(data)-1)
    if(patt(i) == 0)
      result(i)=0.5*(result(i+1)+result(i-1));
    end
  end
end
% графики результатов расчётов
figure;
subplot(2,1,1); stairs(data);
title('Исходный сигнал (data)')
subplot(2,1,2); stairs(patt);
title('Паттерн (patt)');
figure;
subplot(2,1,1); stairs(data);
```

```
title('Исходный сигнал (data)')
subplot(2,1,2); stairs(result);
title('Восстановленный сигнал (result)')
%=============================================================
```

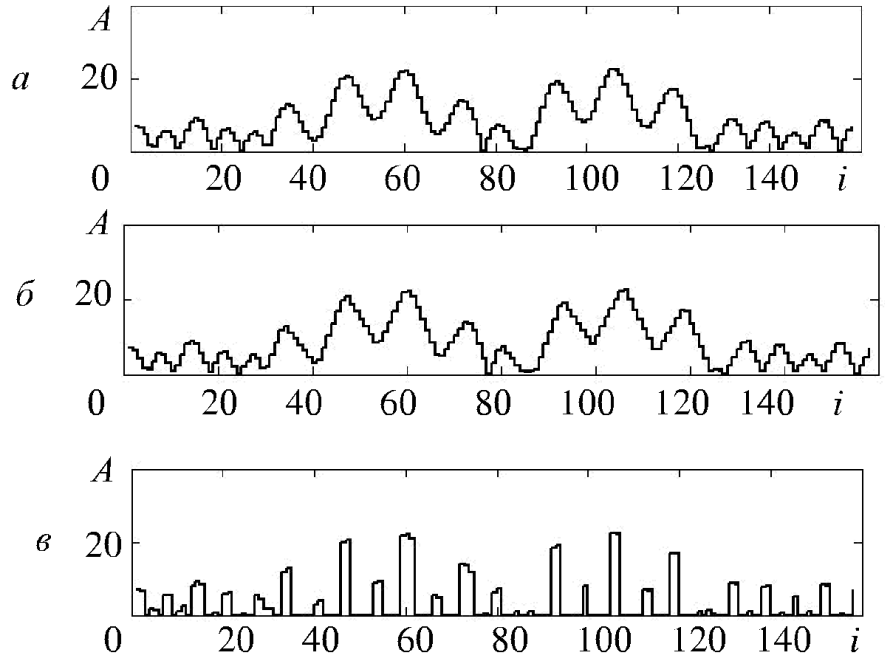


Рис. 7.3. Результаты выполнения программы (листинг 7.2).
а – исходный, *б* – восстановленный сигнал, *в* – паттерн

7.3. Сжатие многомерных цифровых сигналов на базе анализа их дифференциальной структуры

Наиболее часто необходимо применять операции сжатия к так называемому медиаконтенту: статическим и динамическим (видео) изображениям, аудиоданным и др. В последнее время активно развиваются различные форматы видеоданных, на базе которых синтезируются объёмные видеосцены [108, 123]. Цифровые изображения – это один из видов сигналов, к которым можно эффективно применять сжатие на основе анализа дифференциальной структуры. Многие изображения содержат значительные поля градиентных переходов (цвета, яркости), контрастные границы объектов (статические изображения – границы в пространстве, динамические – во времени и в пространстве). Соседние кадры видеопоследовательностей обычно различаются незначительно. Границы объектов, содержащихся в изображениях, можно рассматривать как краевые условия, а поля плавных переходов – как области, удовлетворяющие выбранному анализирующему дифференциальному уравнению.

В качестве примера рассмотрим все этапы сжатия и восстановления статического изображения при помощи анализа дифференциальной структуры [38, 39].

Сжатие изображения:

– выбор дифференциального анализирующего уравнения (или системы) и цветового пространства для формирования паттерна;

– при необходимости – преобразование цветового пространства;

– формирование паттерна (граничных условий);

– сжатие паттерна без потери информации и формирование дополнительных данных для восстановления сигнала.

Восстановление изображения:

– декомпрессия паттерна и дополнительных данных;

– восстановление изображения с помощью выбранного при сжатии анализирующего дифференциального уравнения (системы) с учётом граничных условий, содержащихся в паттерне;

– при необходимости – преобразование цветового пространства.

Подробно рассмотрим каждый этап. В качестве уравнения (системы) для формирования паттерна и последующего восстановления двумерного сигнала изображения $f[x, y]$, заданного на плоскости $X0Y$, выберем уравнение Лапласа, хорошо описывающее плавные переходы в областях между граничными условиями:

$$\frac{\partial^2 f(x, y)}{\partial x^2} + \frac{\partial^2 f(x, y)}{\partial y^2} = 0. \qquad (7.5)$$

После выбора шаблона дифференцирования для функции $f[x, y]$ (рис. 7.4) и перехода к дискретной форме, с учётом шага сетки $\Delta x = \Delta y = 1$ и выражения (7.3), уравнение (7.5) можно записать в виде:

$$\frac{\partial^2 f(x, y)}{\partial x^2} + \frac{\partial^2 f(x, y)}{\partial y^2} \approx$$

$$\approx \frac{f[x + \Delta x, y] - 2f[x, y] + f[x - \Delta x, y]}{\Delta x^2} +$$

$$+ \frac{f[x, y + \Delta y] - 2f[x, y] + f[x, y - \Delta y]}{\Delta y^2} =$$

$$= f[x + \Delta x, y] + f[x - \Delta x, y] +$$

$$+ f[x, y + \Delta y] + f[x, y - \Delta y] - 4f[x, y] = 0, \qquad (7.6)$$

откуда

$$f[x, y] =$$

$$= \frac{1}{4} \left(f[x + \Delta x, y] + f[x - \Delta x, y] + f[x, y + \Delta y] + f[x, y - \Delta y] \right). \qquad (7.7)$$

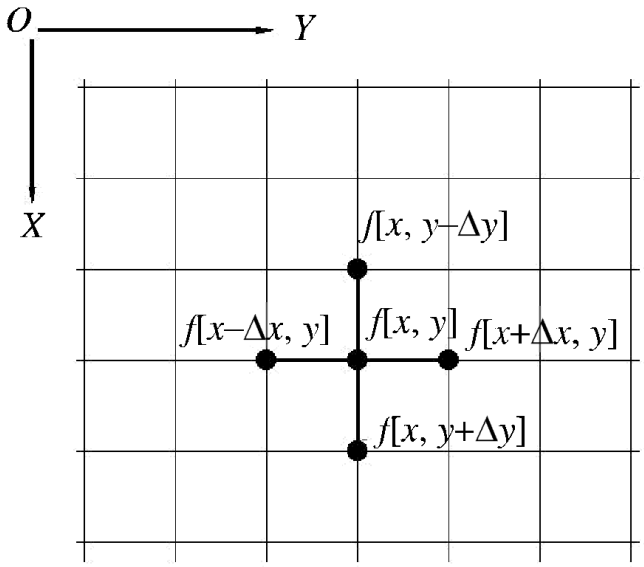


Рис. 7.4. Шаблон дифференцирования функции $f[x, y]$

При использовании предлагаемого метода сжатия выбранный шаблон (см. рис. 7.4) показал достаточно высокую эффективность.

Далее необходимо выбрать цветовое пространство. Практические исследования показали, что наиболее компактный паттерн, и следовательно наиболее высокую степень сжатия, в зависимости от специфики изображения можно получить в цветовом пространстве RGB и YCbCr, см. выражения (1.11) и (1.12). Цветовое пространство RGB адаптировано к особенностям восприятия зрительной системы человека, оно наилучшим образом передаёт специфику цветового градиента, а YCbCr может более эффективно обеспечить сохранение границ объектов изображения при некоторой потере цветовой информации.

Формирование паттерна является ключевым этапом сжатия изображения, от этого зависит качество восстановленного сигнала и в конечном итоге – эффективность сжатия. Отметим некоторые особенности, выявленные при экспериментах с различными подходами к формированию паттерна. Наиболее высокое качество восстановленного сигнала удаётся получить при генерации паттерна, каждому значащему элементу которого ставятся в соответствие все компоненты цветового пространства (т.е. с точки зрения положения элементов граничных условий конфигурация паттернов всех цветовых слоёв одинакова), таким образом, цвет граничных элементов соответствует цвету исходного изображения. Поскольку паттерны всех цветовых компонентов (слоёв) совпадают, возможно сжать последовательность серий не значащих элементов одного слоя и затем использовать его как шаблон для восстановления других слоев, что позволяет существенно повысить фактор сжатия.

В паттерн обязательно включаются граничные пикселы, для статического изображения это две вертикальные строки – справа и слева, и две горизонтальные – сверху и снизу. Для исключения элементов из паттерна сравнивают Δ-критерий и абсолютную величину разности фактического значения сигнала в точке $[x, y]$ и вычисленного по (7.7). Причём для каждого компонента цветового пространства можно выбрать значение Δ-критерия и, следовательно, количество исключённой информации о дифференциальной структуре сигнала (число сохранённых значащих элементов в паттерне для каждого цветового компонента одинаковое).

В листинге 7.3 приведён пример кода формирования паттерна. Двумерный массив `data[][]` содержит исходное изображение, каждый элемент массива имеет тип `IImage`, паттерн формируется в массиве `pattern[][]` каждый элемент массива имеет тип `ColorSpace`. Изначально в `pattern[][]` полностью копируется изображение `data[][]`, далее в двух циклах (листинг 7.3) часть элементов исключается (исключением считается приравнивание к нулю). Переменные (`delLo`, `delAv`, `delHi`) содержат значения

Δ-критериев сравнения для исключения элементов из каждого цветового канала и формирования паттерна. Необходимо маскирование (корректировка) изначально нулевых значений, содержащихся в изображении и скопированных в паттерн, для того чтобы отличать их от исключённых из паттерна. Разность фактического значения сигнала в точке $[x, y]$ и вычисленного по (7.7) рассчитывается по формуле:

$$f[x+\Delta x, y]+f[x-\Delta x, y]+f[x, y+\Delta y]+f[x, y-\Delta y]\ -4f[x, y], \qquad (7.8)$$

причём умножение на четыре в программе производится с помощью поразрядного сдвига (на два бита влево), замена операции деления (7.7) на поразрядный сдвиг (7.8) позволяет существенно сократить время вычислений. От величины Δ-критериев зависит число исключённых из паттерна элементов, а значит – степень сжатия изображения. Чем больше Δ-критерий для цветового канала, тем больше значений будет исключено.

Листинг 7.3. Фрагмент программы формирования паттерна полноцветного изображения (C++)

```
//=============================================================
...
// цветовая палитра
// покомпонентное представление, в скобках указаны компоненты
// палитр (RGB) и (YCbCr) соответственно
struct ColorSpace {
  uchar byteLo; // компонент младший байт (Lower)    (R) (Y)
  uchar byteAv; // компонент средний байт (Average) (G) (Cb)
  uchar byteHi; // компонент старший байт (Hight)   (B) (Cr)
};
// структура для сохранения цветов в виде int
```

```
// один условный "байт" соответствует переменной типа int
struct IImage {
  int iLo; // компонент младший "байт" (Lower)   (R) (Y)
  int iAv; // компонент средний "байт" (Average) (G) (Cb)
  int iHi; // компонент старший "байт" (Hight)   (B) (Cr)
};
...
for (iy=1; iy<Ysize; iy++) {
  for (ix=1; ix<Xsize; ix++) {
    ...
    // расчёт разностных элементов для сравнения с критерием
    int sLo=abs(data[iy-1][ix].iLo+data[iy+1][ix].iLo+
      data[iy][ix-1].iLo+data[iy][ix+1].iLo-
(data[iy][ix].iLo<<2));
    int sAv=abs(data[iy-1][ix].iAv+data[iy+1][ix].iAv+
      data[iy][ix-1].iAv+data[iy][ix+1].iAv-
(data[iy][ix].iAv<<2));
    int sHi=abs(data[iy-1][ix].iHi+data[iy+1][ix].iHi+
      data[iy][ix-1].iHi+data[iy][ix+1].iHi-
(data[iy][ix].iHi<<2));

    // сравнение разностных элементов с критериями
    // исключение из паттерна
    if (sLo<=delLo && sAv<=delAv && sHi<=delHi) {
      // установка признака исключения элемента из паттерна
      // (не являются краевыми условиями)
      pattern[iy][ix].byteLo=0;
      pattern[iy][ix].byteAv=0;
      pattern[iy][ix].byteHi=0;
    } else {
      // корректировка нулевых элементов паттерна
      if (!pattern[iy][ix].byteLo) {
        pattern[iy][ix].byteLo=1;
      }
      if (!pattern[iy][ix].byteAv) {
        pattern[iy][ix].byteAv=1;
      }
      if (!pattern[iy][ix].byteHi) {
        pattern[iy][ix].byteHi=1;
      }
    }
  }
}
...
//================================================================
```

Если полученный в результате рассмотренного преобразования паттерн содержит значительное число повторяющихся элементов и граничных условий достаточно для восстановления исходного сигнала с заданной точностью, можно говорить об эффективном (с целью сжатия) исключении избыточной информации.

Для сжатия паттерна предлагаются следующие преобразования. Первое ориентировано на сжатие значительного числа повторяющихся элементов паттерна (кодирование длин серий), отмеченных в данной реализации нулевым значением. Как уже было сказано, структура (расположение исключённых элементов) совершенно одинакова для всех цветовых компонентов, следовательно, число последовательных серий исключённых элементов кратно числу цветовых каналов, это учитывается при сжатии серий. Полученные на выходе после кодирования длин серий (например, при помощи кода на базе чисел Фибоначчи) данные могут быть сжаты статистическим методом на основе преобразования Хаффмана или арифметического кодирования.

Восстановление изображения производится в обратной последовательности, начиная с декомпрессии паттерна. После получения массива паттерна его используют в качестве граничных условий для решения методом конечных разностей (МКР) анализирующего дифференциального уравнения (7.5), или другого выбранного при сжатии, в соответствии с разностной схемой (7.7) и шаблоном (рис. 7.4). Итерационную схему решения можно представить уравнением:

$$f^t[x,y]=\frac{1}{4}\ f^{t-1}[x+\Delta x,y]+f^{t-1}[x-\Delta x,y]+$$
$$+f^{t-1}[x,y+\Delta y]+f^{t-1}[x,y-\Delta y]\ , \qquad (7.9)$$

где t – номер итерации. Если сопоставлять начало итерации с началом циклов обхода массива восстанавливаемого изображения, а завершение – с их завершением, то, выбрав в циклах положительное приращение значений счётчиков и используя для последующих расчётов ранее вычисленные значения ($f[x-\Delta x,y], f[x,y-\Delta y]$), уравнение (7.9) можно записать в виде:

$$f^t[x,y]=\frac{1}{4}\ f^{t-1}[x+\Delta x,y]+f^{t}[x-\Delta x,y]+$$
$$+f^{t-1}[x,y+\Delta y]+f^{t}[x,y-\Delta y]\ . \qquad (7.10)$$

Завершить итерационный процесс можно по критерию невязки, вычисляемой, например, как сумма разностей всех значений элементов (без граничных условий) для соседних итераций.

7.4. Повышение эффективности сжатия с учётом специфики цифрового сигнала

Цифровые сигналы очень многообразны, потому разнообразны и методы сжатия. Успешность сжатия цифровых сигналов зависит от того, какой способ использован. Сжатие с использованием анализа дифферен-

циальной структуры может быть наиболее эффективно применено к сигналам, содержащим значительные поля градиентных переходов амплитуд и достаточно контрастные границы полей в области существенного изменения амплитуд. Примерами изображений подобного вида могут быть высококонтрастные фотографии объектов, содержащих чёткие контуры и однотонные или градиентные поля:

– рисованные изображения с контрастными границами объектов;
– некоторые качественные пейзажные фотографии;
– высококонтрастные фотографии космических объектов;
– высококонтрастные снимки дистанционного зондирования Земли;
– текстурные изображения и др.

Проведенные исследования показали, что эффективность сжатия изображений с помощью анализа дифференциальной структуры можно повысить, используя дополнительные преобразования. Анализирующее уравнение Пуассона (6.10), (6.11) позволяет сформировать более компактный, по сравнению с уравнением Лапласа, паттерн, но при этом необходимо определить правую часть уравнения. В качестве правой части можно использовать матрицу определённых значений, сформированную по исходному изображению, разделённому на области, например, 8×8 пикселов. Значения матрицы могут быть получены выбором минимальной амплитуды области, этот подход в большинстве случаев позволяет корректировать плотность распределения значений амплитуд таким образом, что к паттерну можно более эффективно применять статистическое сжатие.

При сжатии фотоизображений можно применять специальную обработку для удаления шума, возникающего, например, из-за дефектов ПЗС-матрицы.

Применение низкочастотных фильтров к изображению позволяет исключить из паттерна случайный шум, свойственный цифровым фотографиям, однако при этом возможна потеря высокочастотной полезной информации, например, мелких деталей изображения; в некоторых случаях положительный эффект дает использование высокочастотных фильтров. Следует отметить, что в качестве граничных условий в паттерн сохраняются элементы исходного неотфильтрованного изображения. Фильтрация может быть выполнена с помощью свертки с ФРТ небольшого размера (3×3, 5×5, 7×7), при больших размерах ФРТ эффективен фильтр, использующий операцию «быстрая свертка, основанная на быстром фурье-преобразовании».

В некоторых случаях повысить эффективность сжатия позволяет использование при анализе дифференциальной структуры производных высших порядков и различных шаблонов дифференцирования. Выбор порядка и вида дифференциального уравнения в значительной мере зависит от особенностей изображения. Отображения многих реальных и синтезированных

объектов хорошо анализируются с помощью производных второго порядка, т.е. уравнений Лапласа или Пуассона.

В области сигнала, где значение второй производной мало, возможна потеря полезной информации (рис. 7.5). Наиболее подвержены потере информации области, в которых плавно изменяется функция сигнала, особенно критичным это может быть для областей, где меняется знак первой производной. Избежать потери позволяет включение в паттерн локальных минимумов и максимумов сигнала.

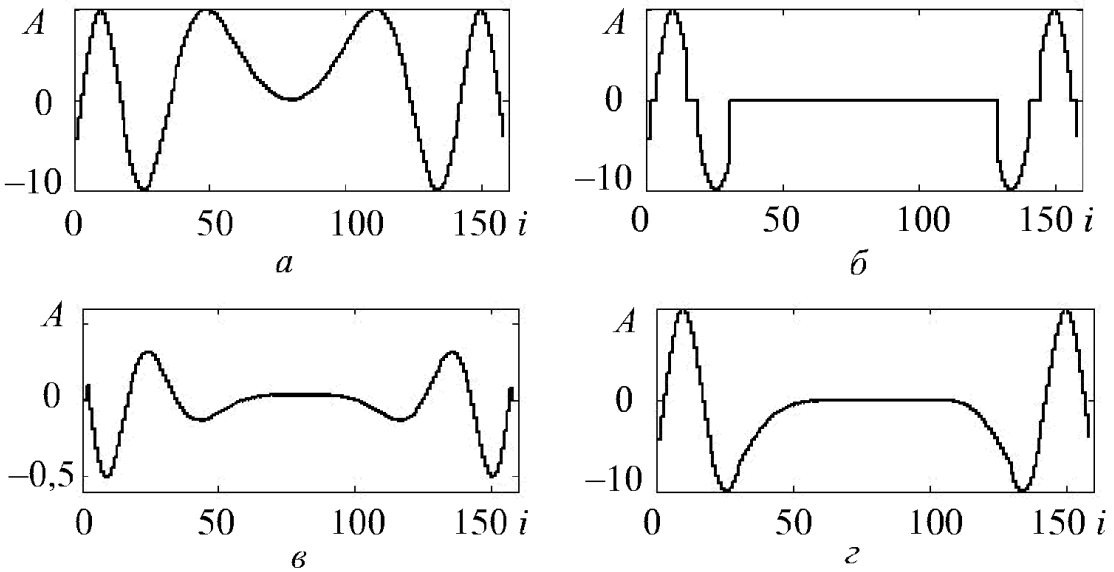


Рис. 7.5. Потеря информации о сигнале при малых значениях второй производной: *а* – исходный сигнал, *б* – паттерн, *в* – вторая производная, *г* – восстановленный сигнал

В листинге 7.4 приведен пример кода программы формирования паттерна с применением операции на базе свертки с ФРТ (3×3), в качестве примера задана ФРТ скользящего среднего, подавляющая высокие частоты сигнала (в том числе случайные шумы) и усиливающая сигнал до 9 раз. Применение низкочастотного фильтра дает хорошие результаты формирования паттерна для последующего сжатия цифровых фотографий, содержащих контрастные объекты и однотонные поля со случайными шумами. Обратим внимание, что в паттерн копируются не результаты свёртки изображения, а элементы исходного изображения, что позволяет оставить больше исходной информации и получить более приемлемые результаты при восстановлении. Свёртка с заданной ФРТ приводит к размытию полей и границ, что, в свою очередь, позволяет учесть в паттерне не только элементы самих границ, но и соседние, удалённые от границ, элементы. Элементы вне границы могут отличаться от граничных, в качестве граничных необходимо выбирать соседние элементы, близкие по значениям к элементам области, и пиковые значения границы. В результате выполнения свёрт-

ки сигнал усиливается, поэтому необходимо нормировать значения элементов сигнала или изменить Δ-критерии. Нормирование динамического диапазона или использование нецелочисленных коэффициентов ФРТ вызовет увеличение времени вычислений, при ограниченных размере и амплитуде ФРТ более эффективно изменять значения Δ-критериев (`delLo`, `delAv`, `delHi`).

Листинг 7.4. Фрагмент программы формирования паттерна с предварительной свёрткой полноцветного изображения (C++)

```
//=================================================================
...
// цветовая палитра
// покомпонентное представление, в скобках указаны компоненты
// палитр (RGB) и (YCbCr) соответственно
struct ColorSpace {
  uchar byteLo; // компонент младший байт (Lower)    (R) (Y)
  uchar byteAv; // компонент средний байт (Average) (G) (Cb)
  uchar byteHi; // компонент старший байт (Hight)    (B) (Cr)
};
// структура для сохранения цветов в виде int
// один условный "байт" соответствует переменной типа int
struct IImage {
  int iLo; // компонент младший "байт" (Lower)    (R) (Y)
  int iAv; // компонент средний "байт" (Average) (G) (Cb)
  int iHi; // компонент старший "байт" (Hight)    (B) (Cr)
};
...
uint jmax=Xsize-1;
uint imax=Ysize-1;
uint ix, iy, tix, tiy, nzero; // счётчики
const int psfxy=3;  // размеры функции рассеяния точки

// матрица ФРТ (PSF - point spread function)
int pulse[psfxy][psfxy]={{1, 1, 1},
                         {1, 1, 1},
                         {1, 1, 1}};

// выделение памяти для образца копии изображения
IImage** itemp2d=new IImage*[imax+psfxy];
for (unsigned int i=0; i<imax+psfxy; i++){
  itemp2d[i]=new IImage[jmax+psfxy];
  memset(itemp2d[i], 0, sizeof(IImage)* (jmax+psfxy));
}

// выполнение 2D быстрой свёртки в целочисленной арифметике
for (int iyd=0; iyd<(int)Ysize; iyd++){
  for (int ixd=0; ixd<(int)Xsize; ixd++){
    // проход по импульсу
    for (int iyp=0; iyp<psfxy; iyp++){
      iy=iyd+iyp;
      for (int ixp=0; ixp<psfxy; ixp++){
```

```
        ix=ixd+ixp;

itemp2d[iy][ix].iLo+=pulse[iyp][ixp]*(int)data[iyd][ixd].byteLo;

itemp2d[iy][ix].iAv+=pulse[iyp][ixp]*(int)data[iyd][ixd].byteAv;

itemp2d[iy][ix].iHi+=pulse[iyp][ixp]*(int)data[iyd][ixd].byteHi;
      }
    }
  }
}

// формирование паттерна
// копирование угловых точек с заменой нулей на единицы
swap(pattern[0][0], data[0][0]);
swap(pattern[0][jmax], data[0][jmax]);
swap(pattern[imax][0], data[imax][0]);
swap(pattern[imax][jmax], data[imax][jmax]);

for (iy=1; iy<imax; iy++){
  // координата по y для массива результат свёртки itemp2d
  tiy=iy+(psfxy>>1);
  // копирование крайних столбцов
  // с заменой нулей на единицы
  swap(pattern[iy][0], data[iy][0]);
  swap(pattern[iy][jmax], data[iy][jmax]);

  for (ix=1; ix<jmax; ix++){
    // координата по x для массива результат свёртки itemp2d
    tix=ix+(psfxy>>1);
    if (iy==1){
      // на первом проходе копирование крайних строк
      // с заменой нулей на единицы
      swap(pattern[0][ix], data[0][ix]);
      swap(pattern[imax][ix], data[imax][ix]);
    }

    int tLo=abs(itemp2d[tiy-1][tix].iLo+itemp2d[tiy+1][tix].iLo+
          itemp2d[tiy][tix-1].iLo+itemp2d[tiy][tix+1].iLo-
          (itemp2d[tiy][tix].iLo<<2));
    int tAv=abs(itemp2d[tiy-1][tix].iAv+itemp2d[tiy+1][tix].iAv+
          itemp2d[tiy][tix-1].iAv+itemp2d[tiy][tix+1].iAv-
          (itemp2d[tiy][tix].iAv<<2));
    int tHi=abs(itemp2d[tiy-1][tix].iHi+itemp2d[tiy+1][tix].iHi+
          itemp2d[tiy][tix-1].iHi+itemp2d[tiy][tix+1].iHi-
          (itemp2d[tiy][tix].iHi<<2));

    // обработка внутренних элементов
    // сравнение с Δ-критериями по каждому цветовому каналу
    if ((tLo<=delLo && tAv<=delAv && tHi<=delHi)){
      // установка признака исключения элемента из паттерна
      // (не являются краевыми условиями)
      pattern[iy][ix].byteLo=0;
```

```
      pattern[iy][ix].byteAv=0;
      pattern[iy][ix].byteHi=0;
      nzero+=3; // счётчик исключённых из паттерна элементов
    } else {
      // элементы паттерна с нулевыми значениями, корректировка
      if (!pattern[iy][ix].byteLo){
            pattern[iy][ix].byteLo=1;
      }
      if (!pattern[iy][ix].byteAv){
            pattern[iy][ix].byteAv=1;
      }
      if (!pattern[iy][ix].byteHi){
            pattern[iy][ix].byteHi=1;
      }
    }
  }
}

// очистка памяти
if (itemp2d && sizeY()){
  for (ix=0; ix<imax+psfxy; ix++)
    delete[]itemp2d[ix];
  delete[]itemp2d;
  itemp2d=0;
}
...
// Функция swap
// копирование точки (в трёхцветной палитре с заменой нулей на
единицы)
inline void swap(ColorSpace& receiver, ColorSpace& source){
  // выполнение проверки, при необходимости замена нулей единицей
  receiver=source; // копирование
  if (!receiver.byteLo){
    receiver.byteLo=1;
  }
  if (!receiver.byteAv){
    receiver.byteAv=1;
  }
  if (!receiver.byteHi){
    receiver.byteHi=1;
  }
}
...
//=================================================================
```

7.5. Вычислительная оптимизация МКР путем отыскания промежуточного решения

При восстановлении сжатых данных немаловажным фактором является скорость. МКР – достаточно медленный итерационный способ решения дифференциальных уравнений, особенно значительное время может

понадобиться для решения многомерных задач. Поэтому необходимо оптимизировать вычисления.

Известен способ сокращения времени вычислений МКР для мягких дифференциальных уравнений [11] в некоторой области **U** с краевыми (граничными для физических областей и начальными для времени) условиями в областях **B**, сущность которого состоит в укрупнении шага h сетки в области **U**. Вследствие этого происходит потеря точности решения вплоть до того, что оно становится непригодным. Например, в основной задаче электростатики могут быть построены модели, в которых заряды рассматриваются как точки, расположенные в узлах сетки. При укрупнении сетки такие граничные условия могут быть потеряны, что принципиально меняет сущность промежуточного решения. При формировании паттерна могут возникать краевые условия, например, в виде уединённой точки, таким образом, для наших задач потеря точности в результате укрупнения сетки также актуальна.

По этой причине распространен подход с последовательным использованием нескольких сеток. Сетка U_1 с самым большим шагом позволяет получить приближенное решение, которое затем уточняют, последовательно применяя сетки с меньшими значениями h (общее число различных сеток обычно 2–3). Шаг сетки может варьировать в зависимости от требуемой точности p вычислений. Таким образом, согласно теореме о сходимости разностного решения [118], вычислительная оптимизация возможна за счет последовательного формирования некоторого промежуточного решения. При этом каждое уточнение решения является итерационным, имеющим вычислительную сложность

$$O\left(t\prod_{i=1}^{n}N_i\right), \tag{7.11}$$

где n – размерность рассматриваемой задачи; N_i – число узлов сетки в каждом направлении; t – число итераций, обеспечивающих требуемую точность на данном этапе. В многомерных задачах вычислительная сложность может быть очень велика даже при разреженной сетке.

Задачу нахождения промежуточного (приближённого) решения в области **U** можно решить путем аппроксимации значений $f(R^n)$ уравнения (7.1) различными функциями, в зависимости от физической сущности задачи, например, многочленом вида:

$$a_0 + a_1 z + a_2 z^2 + ... + a_p z^p, \tag{7.12}$$

где $a_0, a_1, a_2, ..., a_p$ – коэффициенты многочлена; z – переменная, относительно которой происходит аппроксимация.

В общем случае, если линии сетки не параллельны координатным осям, образующим пространство $\mathbf{W}$, то z может не входить во множество переменных исходной задачи. В этом случае необходимо учитывать поворот системы координат, в которых рассмотрен аргумент z, относительно исходной. В случае n-мерной ($n > 1$) задачи аппроксимирование необходимо производить последовательно для всех линий, образующих сетку в каждом направлении, с последующей оценкой средних значений для каждого узла сетки, вычисляемых как средние аппроксимированные значения в точках линий сетки, принадлежащих данному узлу. В определенном смысле можно говорить, что для нахождения некоторого промежуточного решения задача МКР разбивается методом конечных элементов, число которых равно числу линий сетки МКР.

Возможно совместное применение способов укрупнения сетки и нахождения промежуточного решения за счет аппроксимации, ведь даже для укрупненной сетки задача ускорения сходимости остается актуальной. Предложенный способ следует применять с учетом физической сущности процессов и условий каждой конкретной задачи. При изменении шага сетки можно использовать некоторые усредненные граничные условия.

В некоторых случаях (в зависимости от особенностей граничных условий) промежуточное решение, полученное предлагаемым способом [50, 51], может быть не очень «гладким», избежать этого позволит, например, учет (с некоторыми весовыми коэффициентами) вычисленных значений в соседних узлах аппроксимации.

Предлагаемый способ нахождения предварительного решения базируется не на последовательных итерационных приближениях, а на аппроксимации промежуточного решения в области $\mathbf{U}$ по имеющимся данным о граничных условиях $\mathbf{B}$ с учетом правой части (7.1). Вычислительная сложность решения может быть оценена с помощью выражения (7.11), при этом значение t будет существенно меньше, это позволяет говорить о снижении вычислительной сложности предлагаемого способа.

Способ тестировался на различных задачах электротехники, описываемых уравнением Лапласа и Пуассона в двух- и трехмерных пространствах:

– основная задача электростатики (в том числе с зарядами, описываемыми единственным граничным элементом);

– расчет поля конденсатора сложной формы;

– расчет магнитного поля в зазоре электрической машины;

– восстановление цифрового сигнала по паттерну.

Во всех случаях полученное промежуточное решение с заданной точностью было получено меньшим числом итераций МКР. Особый выигрыш наблюдается при гладких краевых условиях, вплоть до того, что при

некоторой степени гладкости может быть получено решение только с применением указанного способа без последующего итерационного решения МКР.

На рис. 7.6 проиллюстрировано существенное ускорение сходимости в результате оптимизации предложенным способом (кривая *2*) на примере невязок в решении двумерной (плоскопараллельной) задачи электростатики. Для разбиения области **W** выбиралась прямоугольная сетка, образованная линиями, параллельными координатным осям, с равным во всех направлениях шагом h. В качестве многочлена (7.12) использовалось уравнение прямой $a_0 + a_1 z$, коэффициенты a_0, a_1 рассчитывались по двум элементам краевых условий из **B**.

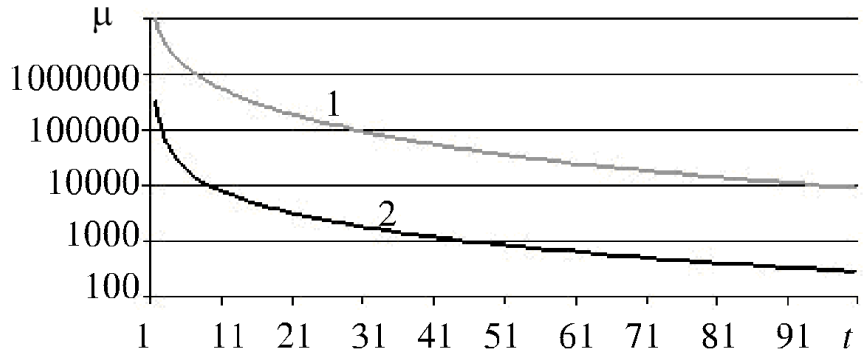


Рис. 7.6. Зависимость значения логарифма суммарных невязок μ по всем узлам сетки от номера итерации

Отметим, что применение рассмотренного способа в некоторых двумерных задачах с целью достижения решения заданной по невязке точности позволило уменьшить число итераций в 10–100 раз, это свидетельствует о существенном повышении эффективности МКР.

7.6. Практические результаты сжатия с помощью анализа дифференциальной структуры

Рассмотрим некоторые результаты практического применения принципа сжатия с помощью анализа дифференциальной структуры на примере цифровых графических изображений. В качестве анализирующего было выбрано уравнение Лапласа (7.5). Изображения для тестирования выбирались из базы данных стандартных изображений [178] с учётом того, что сжатие на основании анализа дифференциальной структуры наиболее эффективно может быть применено к графическим объектам, содержащим контрастные границы и значительные поля градиента или одного тона.

Одно из выбранных для тестирования изображений (рис. 7.7) обладает следующими свойствами: является цифровой фотографией реального физического объекта; содержит контрастные границы объектов; содержит

значительные поля одного тона. Размер исходного файла 196 662 байт, размер изображения 256×256 пикселов, глубина цвета 24 бита, размер цифрового сигнала изображения $256\times256\times3=196\,608$ байт.

Рассмотрим зависимости от значения Δ-критерия: числа исключённых из паттерна элементов; коэффициента сжатия k (6.1); среднеквадратичной ошибки MSE (6.3); пикового отношения сигнал/шум PSNR (6.4); отношения сигнал/шум SNR (6.5). В общем случае можно ожидать закономерности: чем больше Δ-критерий, тем больше элементов исключено из паттерна и тем менее точно можно восстановить исходный сигнал. Когда значение Δ-критерия превысит некоторое (*критическое*), паттерн полноцветного изображения, полученный с помощью программы, приведенной в листинге 7.3, будет содержать только элементы, принадлежащие границе изображения. Следовательно, при критической величине Δ-критерия можно получить максимальное сжатие. При этом вопрос о качестве восстановленного изображения остаётся открытым.

Рис. 7.7. Стандартное тестовое изображение «4.1.05.bmp»

В табл. 7.1 приведены численные результаты экспериментов, для всех цветовых каналов было выбрано одинаковое значение Δ-критерия, паттерн формировался в RGB-пространстве. Для восстановления изображения применялся МКР (1000 итераций) с предварительным отысканием промежуточного решения. Время формирования паттерна (ОС Windows, C++) 31 250 мкс, время восстановления – 1 343 750 мкс. Коэффициент и фактор сжатия рассчитывались путём сопоставления размера файла исходного

изображения «4.1.05.bmp» и размера полученного в результате сжатия с помощью анализа дифференциальной структуры файла. Помимо самих изображений файлы обычно содержат некоторую служебную информацию (время создания, цветовая схема, размеры и т.д.). Длина служебных данных в сжатых файлах 52 байта, в исходном файле «4.1.05.bmp» 54 байта.

Таблица 7.1. Некоторые оценки сжатия изображения «4.1.05.bmp»

Δ-критерий	Исключённых элементов		k	c	MSE	PSNR	SNR
	байтов	%					
0	60	0.030518	0.889	1.125	0.001302	76.98	72.37041
5	23535	11.97052	0.840	1.191	0.121638	57.28	52.66611
10	57828	29.41284	0.699	1.431	0.800832	49.10	44.48139
15	84129	42.79022	0.582	1.719	2.204839	44.70	40.08303
20	107364	54.60815	0.481	2.080	5.200922	40.97	36.356
25	127779	64.99176	0.388	2.578	11.78631	37.42	32.80302
30	144492	73.49243	0.304	3.290	22.84199	34.54	29.92946
35	157290	80.00183	0.234	4.275	36.16757	32.55	27.93361
40	166086	84.47571	0.184	5.436	56.36549	30.62	26.00667
45	171954	87.46033	0.148	6.759	85.77358	28.80	24.18327
50	176058	89.54773	0.123	8.132	127.981	27.06	22.44535
55	179193	91.14227	0.104	9.618	192.4735	25.29	20.67309
60	181608	92.37061	0.089	11.239	268.6562	23.84	19.22483
65	183483	93.32428	0.077	12.991	384.8277	22.28	17.66414
70	184998	94.09485	0.068	14.710	503.2751	21.11	16.49875

На рис. 7.8 приведены полученные при сжатии паттерны и восстановленные изображения.

Обычно для изображений приемлемого качества PSNR ≈20–40 [105]. Сопоставляя табл. 7.1 и рис. 7.8, можно заметить, что при $\Delta \approx 40$–50 наблюдается существенное для визуального восприятия ухудшение качества изображения, возникают области размытия вблизи границ объектов, при этом для $\Delta = 40$ $\mathrm{PSNR} = 30.62$, для $\Delta = 50$ – 27.06. Коэффициент сжатия $k = 0.184$ (для $\Delta = 40$) и $k = 0.123$ (для $\Delta = 50$). При дальнейшем росте значения Δ-критерия наблюдается снижение качества, особенно проявляющееся на границах графических объектов. Изучение структуры паттерна показывает, что снижается и качество отображения текстуроподобной однотонной области кирпичных стен: сравнив рис. 7.8, *и* и *м*, можно заметить, что пропадают граничные условия, соответствующие мелкотекстурным элементам. При Δ =70–80 изображенные предметы, несмотря на значительные искажения, оставались узнаваемыми.

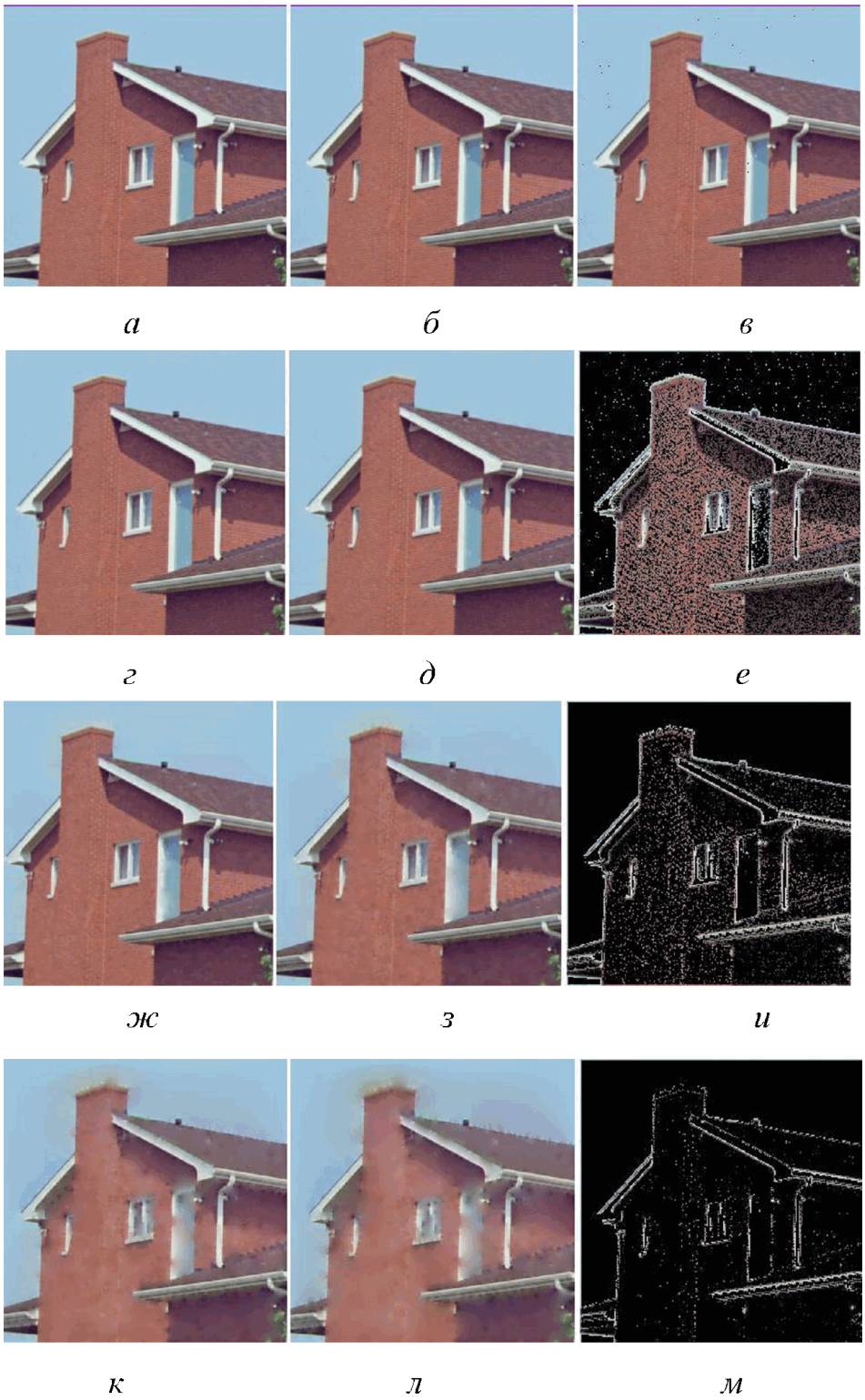


Рис. 7.8. Исходное изображение (*а*); изображения, восстановленные МКР с использованием промежуточного решения (*б, г, д, ж, з, к, л*); паттерны (*в, е, и, м*). Значение Δ-критерия: *б, в* – 0; *г* – 10; *д, е* – 20; *ж* – 30; *з, и* – 40; *к* – 50; *л, м* – 60

Рассмотрим результаты сжатия, полученные с помощью наиболее распространённых методов (табл. 7.2–7.4). Сжатие в частотной области

методом JPEG [84] основано на дискретно-косинусном преобразовании, визуальное качество полученного изображения оценивается по шкале от 0 до 12. Формат PNG (Portable Network Graphik) базируется на сокращении числа разрядов для задания цветового пространства, а следовательно и цветовой палитры, и последующем статистическом сжатии изображения [105]. В отличие от PNG формат GIF позволяет работать с динамическими изображениями.

Таблица 7.2. Некоторые оценки JPEG-сжатия изображения «4.1.05.bmp»

Качество	k	c	MSE	PSNR	SNR
12	0.494	2.023	1.56	46.17	41.56
11	0.359	2.778	5.63	40.62	36.01
10	0.282	3.538	9.35	38.41	33.80
9	0.245	4.069	12.97	37.00	32.39
8	0.222	4.497	17.13	35.79	31.18
7	0.201	4.970	23.31	34.45	29.84
6	0.199	5.006	40.10	32.10	27.49
5	0.187	5.333	46.79	31.43	26.82
4	0.179	5.556	51.48	31.01	26.40
3	0.174	5.746	58.72	30.44	25.83
2	0.164	6.064	75.88	29.33	24.72
1	0.160	6.243	96.81	28.27	23.66
0	0.158	6.316	105.61	27.89	23.28

Таблица 7.3. Некоторые оценки PNG-сжатия изображения «4.1.05.bmp»

Число разрядов	Число цветов	k	c	MSE	PSNR	SNR
24	16777216	0.609136	1.641668	0	–	–
8	256	0.191079	5.233434	10.80395	37.79498	33.18098
7	128	0.15489	6.45619	16.1552	36.04768	31.43368
6	64	0.111638	8.957504	24.60652	34.22031	29.6063
5	32	0.091278	10.95549	34.69823	32.72773	28.11373
4	16	0.040465	24.71249	50.41061	31.10559	26.49158
3	8	0.026284	38.04643	140.4574	26.65536	22.04136
2	4	0.01436	69.63952	252.7536	24.10383	19.48983
1	2	0.007144	139.973	569.2647	20.57766	15.96366

Таблица 7.4. Некоторые оценки GIF-сжатия изображения «4.1.05.bmp»

Число цветов	k	c	MSE	PSNR	SNR
128	0.171243	5.839653	16.91316	35.84856	31.23455
64	0.123583	8.091754	26.3118	33.9293	29.3153
32	0.100335	9.966653	37.71211	32.36599	27.75199

Выберем по одному полученному различными способами (табл. 7.5) сжатия (при минимальном значении коэффициента сжатия) изображению *визуально* приемлемого качества (рис. 7.9). Несмотря на субъективность, такой подход к выбору позволяет учесть специфику восприятия изображения человеком.

а *б* *в*

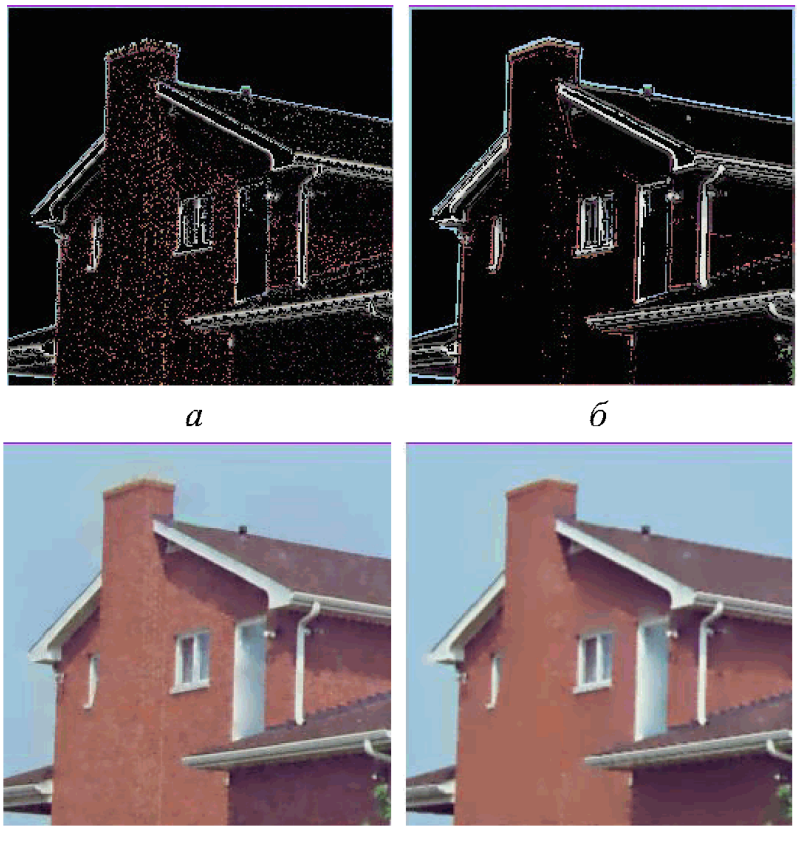
г *д*

Рис. 7.9. Сжатие: *а* – методом анализа дифференциальной структуры ($\Delta = 39$); *б* – методом анализа дифференциальной структуры с предварительной свёрткой; *в* – JPEG (качество 0); *г* – PNG (32 цвета); *д* – GIF (32 цвета)

Таблица 7.5. Результаты сжатия изображения «4.1.05.bmp»

Метод	k	c	MSE	PSNR	SNR
Анализ дифференциальной структуры	0.193	5.183	52.1964	30.95	26.34
Анализ дифференциальной структуры с предварительной свёрткой	0.156	6.391	76.36	29.30	24.69
JPEG	0.158	6.316	105.61	27.89	23.28
PNG	0.091	10.955	34.69	32.72	28.11
GIF	0.100	9.967	37.712	32.37	27.75

Хорошие практические результаты дало применение свёртки перед формированием паттерна (рис. 7.10, табл. 7.6). Для низкочастотной фильтрации с усилением в качестве ФРТ использовалась (3×3)-матрица, заполненная единицами. Такой фильтр позволяет эффективно выделять границы крупных объектов, практически исключая высокочастотную шумовую составляющую и обеспечивая формирование более компактного паттерна. Однако при использовании низкочастотной фильтрации возможна потеря высокочастотных элементов, например, небольших объектов, тонких линий и т.д. Применение высокочастотных фильтров, наоборот, позволяет в восстановленном сигнале сохранить высокочастотные элементы, размыв границы крупных объектов. Таким образом, специфика решаемой задачи определяет выбор вида фильтра.



а *б*

в *г*

Рис. 7.10. Паттерны (*а* – без применения; *б* – с применением свёртки) и восстановленные изображения (*в* – по паттерну *а*; *г* – по *б*)

Таблица 7.6. Оценка влияния низкочастотного фильтра

Свёртка	Число исключённых из паттерна элементов	*k*	*c*	MSE	PSNR	SNR
Нет	166086	0.184	5.434	56.114	30.640	26.026
Есть	166119	0.172	5.829	66.427	29.907	25.293

Полученные результаты показывают конкурентоспособность предложенного метода сжатия с помощью анализа дифференциальной структуры. Дополнительно повысить эффективность сжатия можно, применив арифметическое кодирование вместо использованного нами кода Хаффмана, уравнение Пуассона – вместо уравнения Лапласа. Отметим, что предложенный метод может быть реализован по принципам параллельных вычислений, например, технологии GPGPU. С учётом сказанного способ имеет определённый потенциал развития и перспективы практического применения.

8. Распознавание образов

Постоянно растущий поток данных самой различной природы не может быть эффективно обработан без применения вычислительных средств. Машинная обработка сигналов требуется в различных циклах производства и проектирования, научных исследованиях, военно-стратегических приложениях. Одной из наиболее сложных и актуальных задач цифровой обработки является распознавание образов. Проблема распознавания образов охватывает более широкий круг задач, чем классификация объектов на графических изображениях.

8.1. Задача распознавания образов

Образ (некоторые авторы используют слово *класс*) является привычной единицей мышления, и в общем случае он сложно формализуем, обладает некоторыми свойствами (признаками); методы, доступные какому-либо образу, возможно выделить в отдельную категорию. Характерным свойством образа является наследование, т.е. образ может относиться к некоторым другим образам как родитель или потомок, передавая или наследуя их свойства. Следовательно, в общем случае *задача распознавания* сводится к выделению свойств некоторого объекта и их анализу на предмет принадлежности объекта к некоторому образу (классу). Распознавание образов может быть применено к большому кругу задач. Например, в промышленности актуально распознавание образов некоторых изделий и (или) их элементов при автоматизации; в медицине анамнез пациента можно рассматривать как многомерный сигнал или вектор в многомерном пространстве, характерные особенности которого могут быть использованы для постановки диагноза; в акустике задача распознавания образов – выявление некоторых сигналов или их последовательностей.

Сложность машинной реализации задачи распознавания образов обусловлена отсутствием чёткого определения фундаментального понятия множества. По определению Кантора[15], «*множеством* является соединение в некоторое целое M хорошо различимых предметов m нашего созерцания или нашего мышления» [72]. Невозможность в общем случае решения задачи распознавания образов с помощью формальной арифметики целесообразно рассматривать как ограничение в соответствии с парадоксами теории множеств и теоремой Гёделя о неполноте.

Итак, объекты, хорошо различимые нашим сознанием, в большинстве случаев достаточно сложно или невозможно классифицировать доступными современными вычислительными методами. Поэтому для исследова-

---

[15] Георг Фердинанд Людвиг Филипп Кантор (3 марта 1845, Санкт-Петербург – 6 января 1918, Галле, Германия) – математик, наиболее известен как создатель теории множеств.

ния и распознавания образов наряду с обычными методами ЦОС применяют методы нечёткой логики, выделяемые в отдельную категорию как содержащие некоторое подобие биологических систем, к ним относят нейронные сети, генетические алгоритмы. Но даже применение таких методов не позволят решить задачу распознавания образов в общем случае.

Природа исходного сигнала во многом определяет принципы обработки его цифровой формы с целью распознавания образов. Основываясь на модели, предложенной в монографии [33], приведем этапы работы систем распознавания образов с учетом представления и управления данными, структуры машинных алгоритмов и организации вычислительных процессов

1. Аналого-цифровое преобразование или синтез сигнала, смоделированного при помощи вычислительных средств.

2. Предварительная обработка сигнала (фильтрация шума или для графических данных изменение цветового пространства и т.д.).

3. Формирование графического препарата цифрового сигнала, т.е. некоторой совокупности сигналов, более пригодных для дальнейшего анализа. Например, может быть получен спектр сигнала, рассчитаны производные, произведена его сегментация, аффинные преобразования и т.д.

4. Анализ сигнала. На данном этапе путём оценки различных параметров формируются описание и характеристика сигнала. Например, могут быть выявлены некоторые особенности спектра, функциональные зависимости и геометрические структуры, присутствующие в сигнале.

5. Классификация (распознавание) образов, этот этап обычно является завершающим.

В существующих системах распознавания образов несколько этапов могут быть связаны между собой в единый вычислительный комплекс. Например, этапы 3–5 могут быть связаны в вычислительной операции при использовании нейронных сетей, некоторые системы могут выполнять несколько последовательных вычислительных операций, каждая из которых содержит все этапы или их часть.

8.2. Оценки подобия

Статистические методы могут быть использованы на самых различных этапах решения задачи распознавания образов, область их применения зависит от конкретной задачи. *Регрессионный анализ* достаточно распространён как метод оценки степени влияния изменения некоторых векторных величин $\mathbf{x}_1, \mathbf{x}_2, ..., \mathbf{x}_m \in \mathbf{X}$ на характеристику (вектор) $\mathbf{Y}$ наблюдаемого явления или объекта при выявлении схожести, прогнозировании, в различных аналитических исследованиях. Этот метод позволяет выделять из множества $\mathbf{X}$ ведущие факторы влияния на характеристику $\mathbf{Y}$ и строить регрессионные модели. Существуют различные линейные и нелинейные

регрессионные модели, определяемые выбранной функцией зависимости характеристики $\mathbf{Y}$ от факторов $\mathbf{x_1}$, $\mathbf{x_2}$, ..., $\mathbf{x_m}$ [21], например, для $y \in \mathbf{Y}$ и $x_1 \in \mathbf{x_1}, x_2 \in \mathbf{x_2}, ..., x_m \in \mathbf{x_m}$:

– простая линейная регрессия:

$$y = a_0 + a_1 x_1 + \varepsilon; \tag{8.1}$$

– множественная линейная регрессия:

$$y = a_0 + a_1 x_1 + a_2 x_2 + ... + a_m x_m + \varepsilon; \tag{8.2}$$

– полиномиальная регрессия:

$$y = a_0 + a_1 x + a_2 x^2 + ... + a_m x^m + \varepsilon; \tag{8.3}$$

– линейная по параметрам регрессионная модель общего вида:

$$y = a_0 + a_1 \varphi_1(x_1, x_2, ..., x_m) + a_2 \varphi_2(x_1, x_2, ..., x_m) + ... \\ ... + a_m \varphi_m(x_1, x_2, ..., x_m) + \varepsilon; \tag{8.4}$$

– нелинейная по параметрам модель:

$$y = a_0 + a_1 \ln(b_1 x_1) + a_2 e^{b_2 x_2} + ... + a_m \operatorname{tg}(b_m x_m) + \varepsilon, \tag{8.5}$$

где $a_0, a_1, ..., a_m$ и $b_1, ..., b_m$ – *параметры регрессии*; ε – ошибка; $\varphi_1, ..., \varphi_m$ – функции факторов.

На практике множество характеристик сложным образом нелинейно зависит от факторов φ; при определённых допущениях с точностью, приемлемой для решения конкретной задачи, в большинстве случаев удаётся получить линейные регрессионные зависимости. Количественной оценкой линейной зависимости между векторами $\mathbf{X}$ и $\mathbf{Y}$ служит коэффициент корреляции, вычисляемый по формуле:

$$\rho = \frac{\operatorname{cov}\ \mathbf{X}, \mathbf{Y}}{\sigma_X \sigma_Y} = \frac{\sum_{i=1}^{m} \left[x_i - \bar{x} \quad y_i - \bar{y} \right]}{\sqrt{\sum_{i=1}^{m} x_i - \bar{x}^{\,2} \sum_{i=1}^{m} y_i - \bar{y}^{\,2}}}, \tag{8.6}$$

где $\operatorname{cov}\ \mathbf{X}, \mathbf{Y}$ – ковариация векторов $\mathbf{X}$ и $\mathbf{Y}$; σ_X – среднеквадратическое отклонение $\mathbf{X}$; σ_Y – среднеквадратическое отклонение $\mathbf{Y}$; $\bar{x}, \bar{y}$ – математическое ожидание $\mathbf{X}$ и $\mathbf{Y}$.

Корреляцию между двумерными матрицами **X** и **Y** можно вычислить по следующей формуле:

$$\rho = \frac{\sum_{i=i}^{I}\sum_{j=1}^{J}\left[\ x_{ij} - \bar{x} \quad y_{ij} - \bar{y}\ \right]}{\sqrt{\sum_{i=1}^{I}\sum_{j=1}^{J} x_{ij} - \bar{x}\ ^2 \sum_{i=1}^{I}\sum_{j=1}^{J} y_{ij} - \bar{y}\ ^2}}. \qquad (8.7)$$

При распознавании образов в многомерных сигналах можно использовать корреляционные оценки между отдельными фрагментами этих сигналов. В общем случае корреляционными считают все методы, основанные на вычислении оценок подобия анализируемого и эталонного сигналов, например, свёртку. Достаточно хорошие помехоустойчивость и эффективность распознавания сложноструктурированных объектов возможно получить за счет разделения сигнала на фрагменты, оценки корреляции между ними и набором эталонов при последующей классификации [32, 124]. При корреляционном анализе многомерных сигналов необходимо решать многопараметрическую задачу оптимизации с целью выбора размеров и положения фрагментов. С одной стороны, при малых размерах фрагмента корреляционные оценки более устойчивы к повороту относительно эталона и требуют меньших расчётов, с другой – высока вероятность ложного распознавания.

Задачу определения положения фрагментов, подобных эталонным, можно решить с помощью скользящего окна, когда выбранная область (фрагмент) плавно перемещается по всему сигналу, что требует значительного количества вычислений. Дополнительные сложности обусловлены необходимостью соответствия масштабов искомого объекта во фрагменте сигнала и объекта в эталонном сигнале. Наиболее подходящий способ выбирается с учётом специфики условий задачи и требований к ее решению. Например, существенную часть вычислений можно сократить и получить хороший результат при корректировке размеров и положения фрагментов оператором. В большинстве случаев мерой эффективности систем распознавания образов служит уровень их автоматизации, т.е. способности самостоятельно выполнять этапы распознавания – от получения входных данных до выдачи готовых результатов.

Пример программы вычисления с помощью двумерной свёртки вероятного положения искомого образа приведён в листинге 8.1, результаты выполнения программы представлены на рис. 8.1. Если размеры сигнала, содержащего искомый образ, невелики, то свёртка выполняется достаточно быстро, в противном случае можно применять быструю свёртку на основе БПФ.

Листинг 8.1. Определение вероятного положения образа (MATLAB)

```
%================================================================
clear all;        % удаление переменных
close all;        % закрытие всех окон
clc;              % очистка командного окна

% загрузка исследуемого изображения
imgRGB=imread('4.1.05.bmp');
% преобразование исследуемого изображения в градации серого
imgGray=im2double(rgb2gray(imgRGB));

% загрузка файла с сигналом, содержащим искомый образ
testRGB=imread('test.05.bmp');
% преобразование изображения в градации серого
testGray=im2double(rgb2gray(testRGB));

% подготовка препарата искомого образа
% получение размера
sz=size(testGray);
objGray=zeros(sz(1),sz(2));

% зеркальное отражение для вычисления свёртки
objGray(1:end,1:end)=testGray(end:-1:1,end:-1:1);

% нормирование амплитуды приведением к среднему значению, равному
нулю,
% для уменьшения отклика на однотонные области
tM=mean2(objGray);
objGray=objGray-tM;

% выполнение свёртки
cResult=conv2(imgGray,objGray,'same');
% нахождение максимума в матрице свёртки,
% как наиболее вероятного положения искомого образа
[T,y]=max(max(cResult,[],2));
[T,x]=max(max(cResult,[],1));

% визуализация результатов
figure; colormap(Gray);
subplot(2,2,1); imagesc(imgGray);
title('Исследуемое изображение');
subplot(2,2,2); imagesc(testGray);
title('Эталонное изображение (искомый образ)');
subplot(2,2,3); imagesc(objGray);
title('Препарат искомого образа');
subplot(2,2,4); imagesc(cResult);
title(['Вероятное      положение      объекта      X=',num2str(x),'
Y=',num2str(y)]);
text(x,y,'\leftarrow point','FontSize',14)
%================================================================
```

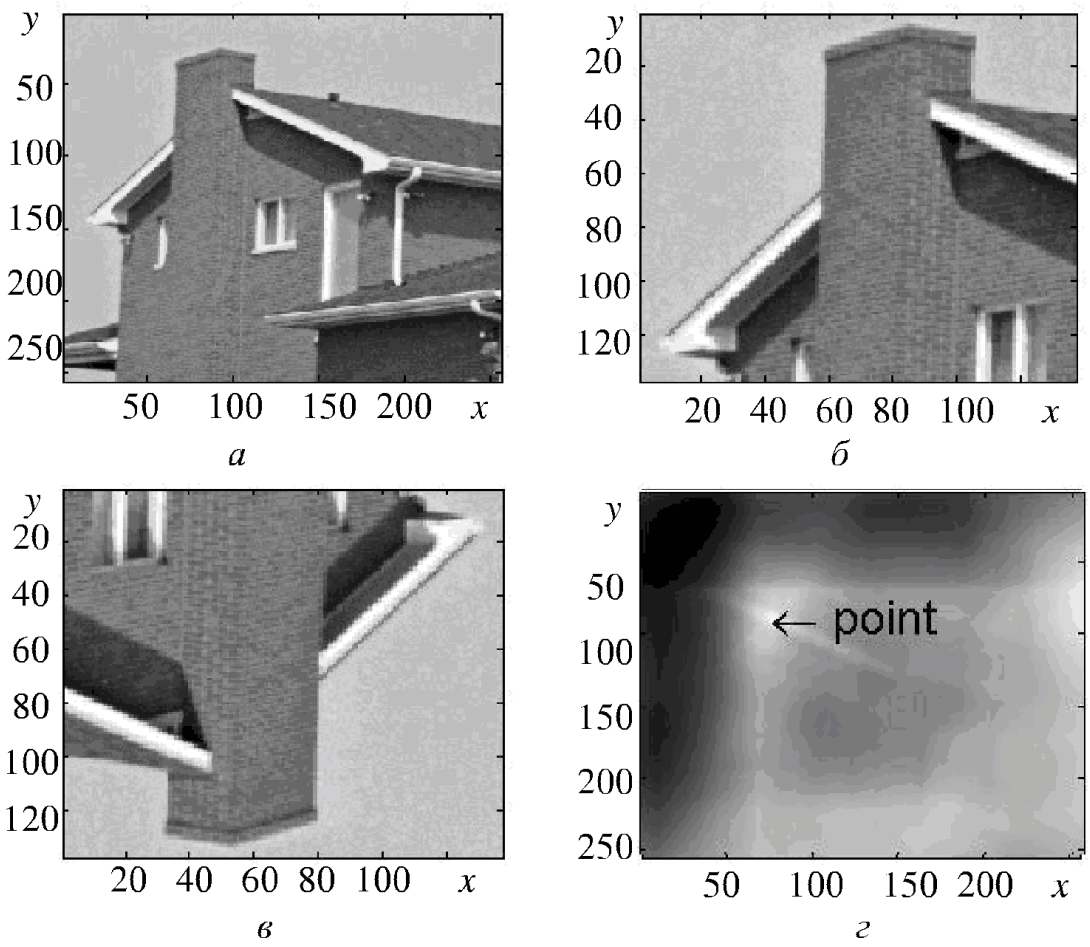


Рис. 8.1. Определение вероятного положения образа
а – исследуемое изображение, *б* – эталонное изображение (искомый образ)
в – препарат искомого образа, *г* – результат свёртки,
вероятное положение объекта (X=77, Y=89)

Оценку подобия фрагмента сигнала и некоторого образа также можно получить, вычислив их среднеквадратическую ошибку.

Рассмотренный способ определения положения образа устойчив к шумам (см. листинг 8.1), сложности могут возникнуть при повороте образа или изменении его масштаба. В этом случае могут потребоваться дополнительные преобразования (поворот, масштабирование) образа и дополнительное вычисление свёртки, что может существенно увеличить время вычислений.

8.3. Анализ спектра для распознавания образов

В ряде случаев отыскание образов необходимо выполнять в частотной, а не пространственной области, например, для распознавания звуковых сигналов частотное представление в виде спектра информативней временной диаграммы. Пример распознавания в частотной области – выявление на изображении объектов некоторых линейных размеров по пространственному

спектру. Для получения пространственного и/или временного спектра могут быть использованы фурье-, вейвлет-преобразование или преобразование по БПВ.

При распознавании образов можно использовать как пространственный и/или временной (для динамического изображения) спектр, определяющий характеристики расположения объектов, так и энергетический спектр, соответствующий энергетическим характеристикам, т.е. собственному и отражённому излучению сигнала.

Для определения истинных энергетических характеристик необходимо учитывать искажения и преобразования, вносимые измерительным прибором. Исследовательский спутник SDO (Solar Dynamics Observatory) обеспечивает наблюдения динамики Солнца преимущественно в области спектра УФ-излучения, не видимого для глаза человека [130, 175]. Визуализация полученных изображений производится при помощи подмены истинного спектра (true color) ложными цветами (false color). Замена энергетического спектра сигнала производится достаточно часто, например, с целью акцентирования внимания на некоторых объектах.

В листинге 8.2 выполнено отображение цветового спектра изображения в пространства RGB и HSV.

Листинг 8.2. Визуальный анализ цветового спектра изображения

```
%================================================================
clear all;      % удаление переменных
close all;      % закрытие всех окон окна

% загрузка и трансформация данных изображения
imgRGB = imread('latest_512_0094.bmp');
imgHSV = rgb2hsv(imgRGB);          % преобразование цветового про-
странства
RGBline = reshape(imgRGB,[],3); % преобразование в двумерный мас-
сив
HSVline = reshape(imgHSV,[],3); % преобразование в двумерный мас-
сив

% визуализация
pview = repmat(5,size(RGBline,1),1);
pcolor = double(RGBline(:,:))./255;

% RGB-модель
figure;
scatter3(RGBline(:,1),  RGBline(:,2),  RGBline(:,3),  pview,  pco-
lor,'filled');
xlabel('R-color');
ylabel('G-color');
zlabel('B-color');

% HSV-модель
figure;
```

```
scatter(HSVline(:,1), HSVline(:,2), pview, pcolor,'filled');
xlabel('H-component');
ylabel('S-component');
%=================================================================
```

На рис. 8.2 визуализирован цветовой спектр фотографии Солнца (http://sdo.gsfc.nasa.gov/), полученной SDO на длине волны 9.4 нм. Графические результаты выполнения программы приведены на рис. 8.3. Визуальный анализ цветовых гистограмм позволяет сделать выводы об интенсивности излучения, наличии вспышек. На диаграмме RGB заметен характерный «хвост», уходящий в область белого цвета (область максимальных значений интенсивности RGB). На диаграмме HSV область вспышек отображена в окрестности значения 0.21 H-компонента. Таким образом, анализ цветового спектра изображения позволяет получить информацию об интенсивности вспышек.

Дополнительную информацию может дать анализ цветового спектра отдельных фрагментов изображения. В некоторых случаях цветовые диаграммы позволяют выявить почти незаметные на исходном изображении объекты.

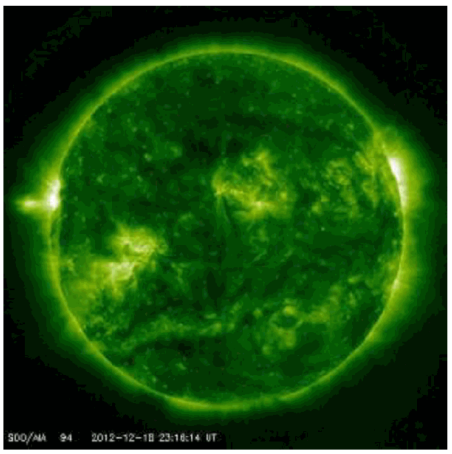


Рис. 8.2. Изображение Солнца (ложные цвета) в УФ-диапазоне, SDO 18 декабря 2012 г., 23:16:14 UT

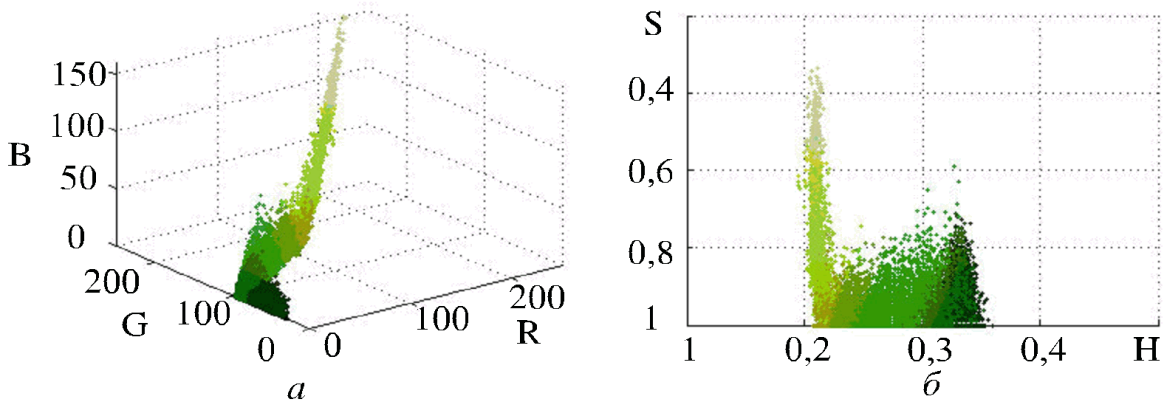


Рис. 8.3. Диаграммы цветового спектра изображения Солнца: *а* – RGB; *б* – HSV

Анализ энергетического спектра изображения при распознавании образов, активно применяют для автоматизированного или автоматического исследования космических фотографий, данных дистанционного зондирования Земли, анализа медицинских изображений и др.

Всесторонний анализ спектра цифрового сигнала позволяет получить значительное количество информации, в том числе для решения задачи распознавания образов.

8.4. Классификация образов

На основании вычисленных оценок подобия возможно классифицировать объекты, выделенные в сигнале. Для этого обычно используют кластерный анализ, позволяющий разделить исходную совокупность объектов на группы схожих, близких (по некоторому множеству критериев) объектов. Выделенные кластеры могут быть полезны для выявления некоторых признаков, например, определённого взаимного расположения и классификации образов в анализируемом сигнале. Разнообразие алгоритмов кластеризации объясняется широким кругом решаемых с их помощью задач и различием подходов к кластерному анализу. Один из подходов состоит в разбиении исходного множества объектов на основе значения целевой функции, параметрами которой являются, например, среднеквадратическая ошибка, коэффициент корреляции и др.

Вычислить сходность данных позволяют:

– евклидова метрика:

$$d_E = \sqrt{\sum_{i=1}^{n} f[i] - s[i]^{\,2}}, \tag{8.8}$$

где $f[i]$, $s[i]$ – векторы;

– взвешенная евклидова метрика:

$$d_{sE} = \sqrt{\sum_{i=1}^{n} p[i]\ f[i] - s[i]^{\,2}}, \tag{8.9}$$

где $p[i]$ – вес (значимость) i-го признака;

– l_m -нормы:

$$l_m = \left[\sum_{i=1}^{n} \left| f[i] - s[i] \right|^m \right]^{\frac{1}{m}}, \tag{8.10}$$

при $m = 1$ l_1 служит манхеттоновской мерой (*Manhatten distance* metric);

– супремум-норма, или расстояние Чебышева[16]:

[16] Пафнутий Львович Чебышев (4 мая 1821, Окатово, Калужская губерния – 26 ноября 1894, Санкт-Петербург) – выдающийся математик и механик.

$$\rho_\infty = \sup_{i=1,2,\ldots n} \left| f[i] - s[i] \right| \ ; \tag{8.11}$$

– дистанция Хемминга – используется номинальная двухпозиционная шкала сравнения схожести символов, стоящих в одинаковых позициях некоторых сигналов $f[i]$ и $s[i]$:

$$\begin{cases} d_H = \dfrac{1}{n}\sum_{i=1}^{n} \varphi \ \ f[i], s[i] \ , \\ f[i] = s[i] \rightarrow \varphi \ \ f[i], s[i] \ = 0, \\ f[i] \neq s[i] \rightarrow \varphi \ \ f[i], s[i] \ = 1. \end{cases} \tag{8.12}$$

Существуют и другие метрики, позволяющие оценивать схожесть и производить классификацию различных данных. В листинге 8.3 рассмотрен пример классификации графических образов при помощи евклидовой метрики.

Листинг 8.3. Классификация графических образов (MATLAB)

```
%=================================================================
clear all;        % удаление переменных
close all;        % закрытие всех окон
clc;              % очистка командного окна

nimg=25; % число изображений объектов с исходным размером 81x81
% выделяем память для массивов
imgObj=zeros(nimg,81,81);
imgH=zeros(nimg,81,81);
metr=zeros(nimg);
x=zeros(1,nimg);
y=zeros(1,nimg);
% дескрипторы окон вывода графических данных
hig=figure;
hic=figure;
hib=figure;
nt=ceil(sqrt(nimg)); % число элементов графического поля
n1=1;                % счётчик

% загружаем массив изображений
for i=1:1:nimg
  imgRGB=imread([num2str(i),'.bmp']);
  imgGray=im2double(rgb2gray(imgRGB));
  imgObj(i,:,:)=imgGray;
  % двумерная функция Гаусса для вычисления свёртки и
  % определения графического "центра тяжести" изображения
  winH=fspecial('gaussian',81,10);
  % свёртка
  cnv=conv2(imgGray,winH,'same');
  % ищем максимум в матрице результате свёртки,
  % как наиболее вероятное положение искомого образа
  [T,px]=max(max(cnv,[],2));
```

```
  [T,py]=max(max(cnv,[],1));
  % запоминание положения графического "центра тяжести"
  x(i)=px;
  y(i)=py;

  % визуализация исходных изображений
  figure(hig);
  subplot(nt,nt,n1); colormap(Gray); imagesc(imgGray);
  title([num2str(n1)]);

  % визуализация результатов свёртки
  figure(hic);
  subplot(nt,nt,n1); colormap(Gray); imagesc(cnv);
  % вывод порядкового номера изображения и
  % координат графического "центра тяжести"
  title([num2str(n1),', (',num2str(px),',',num2str(py),')']);
  % приращение счётчика
  n1=n1+1;
end

% счётчики
n1=1;
n2=1;

% вычисление метрик схожести, кластеризация
for i=1:1:nimg
  % текущее базовое изображение для сравнения,
  % выделенное с учётом "центра тяжести"
  base(:,:)=imgObj(i,x(i)-18:x(i)+18,y(i)-18:y(i)+18);

  % отображение базовых изображений
  figure(hib);
  subplot(nt,nt,n1); colormap(Gray); imagesc(base);
  title([num2str(n1)]);
  % приращение счётчика
  n1=n1+1;
  % цикл вычисления метрик схожести
  for j=i+1:1:nimg
    % очередное изображение для сравнения
    % выделенное с учётом "центра тяжести"
    test(:,:)=imgObj(j,x(j)-18:x(j)+18,y(j)-18:y(j)+18);
    % вычисление евклидовой метрики
    ed=test-base;
    clast(n2)=sqrt(sum(sum(ed.*ed)));
    % приращение счётчика
    n2=n2+1;
  end
end
% построение классификационной дендрограммы
Z=linkage(clast,'average');
figure;
dendrogram(Z);
%=================================================================
```

Вначале выполняются инициализация переменных и резервирование памяти для некоторых данных. Поскольку исходные положения символов имеют достаточно существенный разброс на плоскости изображений (рис. 8.4, цифры сверху – условный номер графического символа), в цикле загрузки происходит определение «центров тяжести» графических образов при помощи свёртки с двумерной функцией распределения Гаусса. По максимальному значению матрицы, являющейся результатом свёртки (рис. 8.5, цифры сверху – условный номер графического символа и координаты максимального значения свёртки), определяется условная точка – «*центр тяжести*» графического объекта. Выделенные центрированные графические объекты представлены на рис. 8.6 (цифры сверху – условный номер графического символа). По осям рис. 8.4–8.6 приведены координаты пространства изображений.

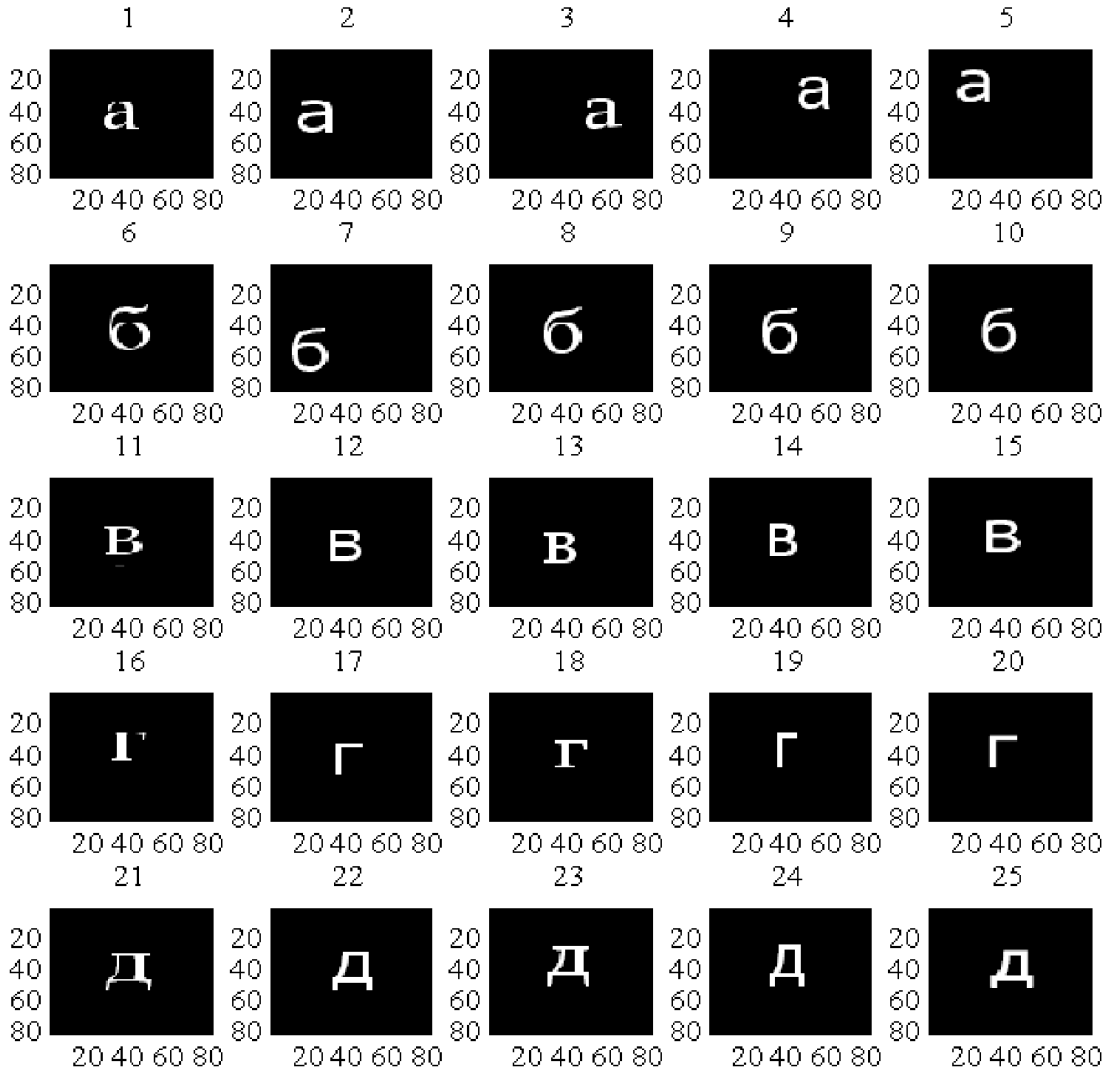


Рис. 8.4. Визуализация исходных изображений

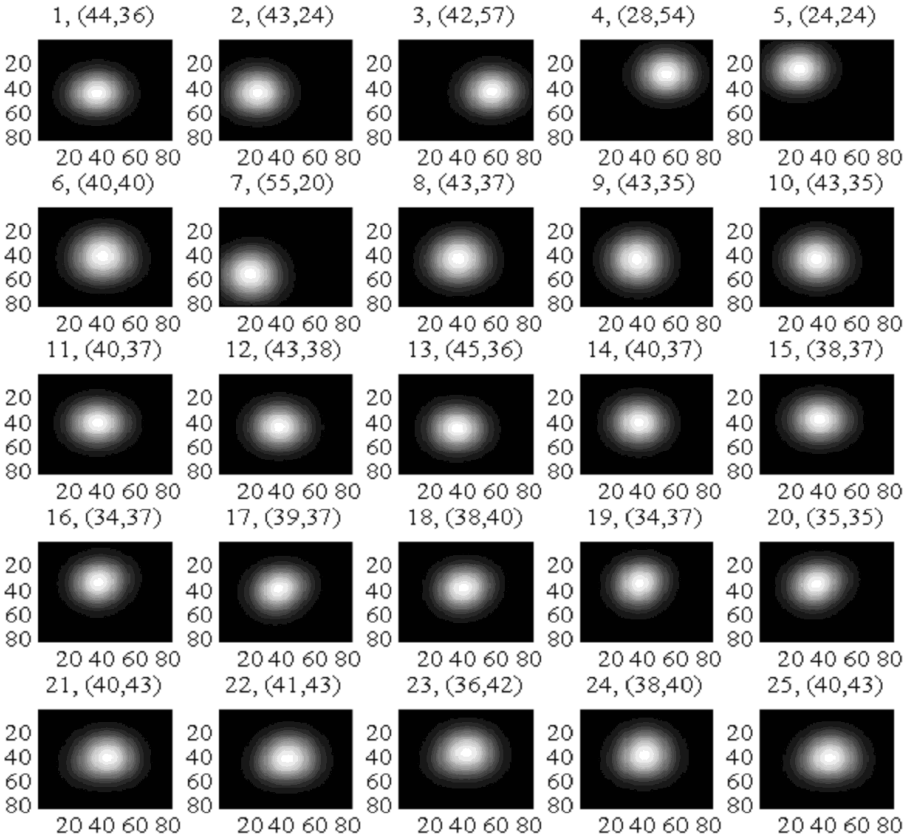


Рис. 8.5. Визуализация результатов свёртки
с целью определения «центров тяжести» графических объектов

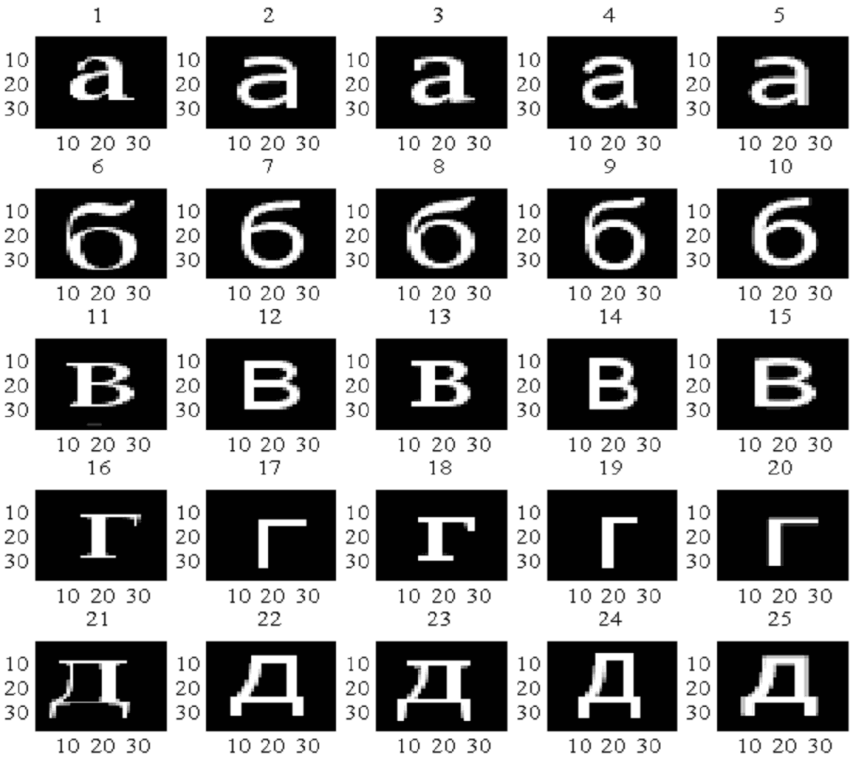


Рис. 8.6. Визуализация выделенных графических объектов
для последующей кластеризации

В соответствии с вычисленной евклидовой метрикой (8.8) осуществляется классификация графических образов. Результаты классификации приведены на рис. 8.7: видно, что все символы корректно распределены по группам (см. рис. 8.4–8.6), исключение составляют 21 и 23 («д»), это связано со значительной вариабельностью начертания символов данной группы. При сравнении расстояний внутри группы и между группами можно сделать вывод об устойчивости алгоритма. Наилучшим образом были классифицированы символы группы «г». Введение дополнительных параметров кластеризации позволяет повысить устойчивость, зачастую применяются дополнительные способы обработки информации, для графических изображений это может быть векторизация, скелетонизация и др.

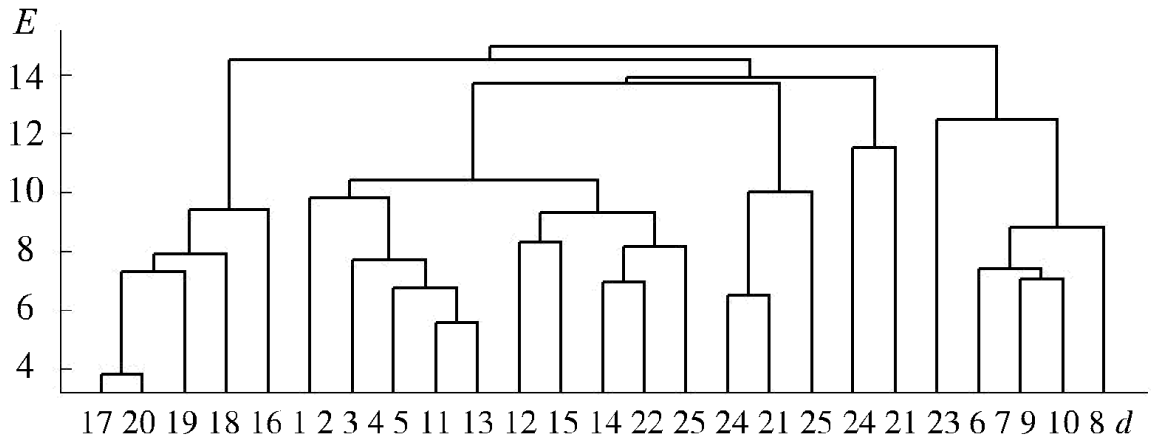


Рис. 8.7. Дендрограмма классификации графических символов
d – условный номер графического символа, *E* – евклидова метрика

8.5. Морфологический анализ

В морфологическом анализе цифровых сигналов можно выделить два этапа распознавания: морфем (элементарных образов) и некоторых морфологических групп, которые могут формировать образы различной сложности. В практической системе распознавания образов анализ может выполняться как от частного к общему (от отдельных морфем к группам) – индуктивный, так и наоборот (от морфологической группы к морфемам) – дедуктивный. Морфемы могут образовывать фрактальные конструкции, если неделимые морфологические элементы на одном уровне содержат морфологические группы другого уровня. Например, амплитуда речевого звукового сигнала представлена на рис. 8.8; видно, что периоды произнесения слов и пауз между словами хорошо различимы. При дедуктивном подходе на некотором этапе распознавания звуковых образов можно выделить периоды произнесения слов и считать, что такие периоды неделимы, т.е. являются морфемами, на следующем этапе детализации возможно анализировать каждый выделенный фрагмент, находя в нём определённые

морфологические элементы их группы и последовательности. На практике для распознавания звуковых образов, например речи, более эффективно работать с сигналом в частотной или в частотной и временной областях. Это обусловлено тем, что при восприятии звука решающее значение имеет амплитуда присутствующей в сигнале частоты, а не фаза.

Рис. 8.8. Временная диаграмма звукового потока речи

Рассмотрим пример индуктивного подхода: в некотором цифровом сигнале идентифицировать достаточно сложный объект по присутствию в нём заранее известной морфемы.

Немаловажным этапом при распознавании образов, в частности при морфологическом анализе, является подготовка морфологических препаратов. Под подготовкой подразумевается приведение элементов цифрового сигнала к виду, более удобному для обработки, например, выделение границ объектов, скелетонизация. Границы объектов позволяет выделять операция дифференцирования, которую можно выполнить с использованием свёртки, например, для двумерного сигнала с помощью функций рассеяния точки вида (4.14)–(4.16).

Схожие (например, по цвету или яркости) и/или соприкасающиеся области могут быть объединены, для этого используется алгоритм заливки с последовательным перебором направлений, обычно реализуемый рекурсивно, но при ограниченных объёмах памяти и значительных областях, подлежащих заливке, – в виде цикла.

При выделении морфологических препаратов также используется *скелетонизация*, т.е. для графического, точечного изображения выделяется некоторый многосвязный (для двумерного изображения – плоский) граф, ему соответствующий [85].

Полученные морфологические препараты классифицируют по некоторым признакам, находя сходства и различия с некоторыми базовыми элементами. В некоторых случаях морфологический анализ позволяет получить существенные результаты при решении задачи распознавания образов.

8.6. Нейронные сети

В системах распознавания образов широко применяются нейронные сети, моделью которых служат биологические системы [24], поскольку единицей восприятия, или мыслительной единицей, живых организмов является образ. Нейронные сети обычно решают комплекс задач, в том числе выделение полезного сигнала из шума, разделение сигнала на некоторые фрагменты (подготовка препарата), непосредственно распознавание и классификация образа. Их основная особенность – способность к обучению, что позволяет формировать алгоритм принятия решения, сложно формализуемый привычными способами. Поэтому вычислительные системы, построенные на базе нейронных сетей, иногда называют интеллектуальными системами.

В силу того что большинство нейронных сетей ориентированы на решение определённого круга задач, существует значительное число их разновидностей. Основным элементом любой нейронной сети является ячейка, моделирующая работу биологического нейрона. Нейрон имеет несколько входов (рис. 8.9), по аналогии с синаптическими волокнами биологического нейрона, и единственный выход – аксон. Сигналы, поступающие на вход нейрона, подаются в качестве аргумента на функцию активации нейрона:

$$y = f\left[\sum_{i=1}^{N} x_i a_i\right], \tag{8.13}$$

где N – число входов (синапсов) нейрона; x_i – сигнал на i-м входе нейрона; a_i – весовой коэффициент i-го входа нейрона; f – функция активации нейрона.

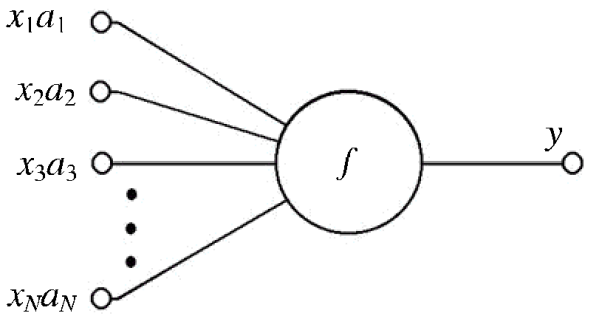


Рис. 8.9. Функциональная структура нейрона

На выходе нейрона получается значение его функции состояния; задача обучения нейрона состоит в выборе весовых коэффициентов, обеспечивающих наилучшее решение. Обучение обычно производится на некоторой, заранее сформированной, выборке, ещё одна выборка обычно используется для оценки качества обучения. Для активации нейрона обычно используется нелинейная функция, например, пороговая, сигмоид и др.

На сегодняшний день выбор оптимальной структуры сети для решения конкретной задачи недостаточно формализован [14, 85]. Помимо того, нейронные сети ориентированы на задачи, решение которых затруднительно другими методами, и именно такие задачи могут иметь нестандартные исходные условия, значительно отличающиеся от обучающих выборок. Качество решения конкретной задачи зависит от обучения нейронной сети, при котором, однако, невозможно учесть существенные отклонения от ограниченной обучающей выборки. Преодолеть эту проблему позволяет увеличение обучающей выборки и размеров нейронной сети, но при этом возникает так называемая проблема «*проклятия размерности*».

Отличительной особенностью нейронных сетей является возможность *интеграции* последовательных этапов распознавания образа.

Практические исследования проблемы распознавания образов и изучение искусственных нейронных сетей способствуют пониманию механизмы работы мозга и высшей нервной деятельности.

ЗАКЛЮЧЕНИЕ

Исследования в области информационных технологий значительно продвинулись с развитием вычислительной техники. Особое значение для человечества имеют развитие сетевых технологий и возможность доступа к огромному объёму информации. В современных условиях доступность информации определяется не только пропускной способностью вычислительных сетей, но и эффективностью методов обработки данных, т.е. анализа, синтеза и моделирования. Таким образом, развитие научных основ построения методов и средств автоматической и автоматизированной обработки изображений в вычислительных системах с применением CALS-технологий и распределённых вычислений при внутренней и межсистемной интероперабельности является актуальным направлением научных исследований в сфере создания систем автоматизированного проектирования (САПР), позволяющих решать научную задачу повышения эффективности использования вычислительных ресурсов. Возможности современных вычислительных систем определяют их «интеллектуальность», заключающуюся в способности к самоорганизации и упорядочиванию информации. Всё это благодаря обратным связям определило развитие не только вычислительной техники и технологии, но и физиологии, психологии личности и социума. Значительный вклад в понимание многих механизмов высшей нервной деятельности человека и животных привнесло исследование нейронных сетей.

Многие аспекты современных информационных технологий, подобно «мирному атому», содержат в себе реальные угрозы для отдельных людей и целых социальных групп. Поэтому хочется видеть цивилизацию, применяющую достижения науки лишь в созидательном направлении, позволяющем развиваться и реализовывать свой творческий потенциал каждому человеку в гармонии с природой и внутреннем миром высокоразвитой личности.

СПИСОК ЛИТЕРАТУРЫ

1. *Аверин Д. В., Лихарев К. К.* Когерентные колебания в туннельных переходах малых размеров // ЖЭТФ. 1986. Т. 90, вып. 2. С. 733–743.

2. *Агафонцев Д. С.* Устойчивость и коллапс солитонов вблизи перехода от мягкой к жесткой бифуркации в системах гидродинамического типа: Дис. канд. физ.-мат. наук. Черноголовка, 2008. 70 с.

3. *Аксентов Ю. В., Джакония В. Е., Жебель Б. Г., Колин К. Т., Кондратьев А. Г.* Телевидение / Под ред. *П.В. Шмакова*. М.: Связь, 1970. 540 с.

4. *Александров Г. Н., Борисов В. В., Каплан Г. С., Кукеков Г. А., Карпенко Л. Н., Кузнецов В. Е., Лунин В. П., Моисеев М. Б., Пирятинский А. В., Соснин В. А., Тонконогов Е. Н., Филиппов Ю. А., Ярмаркин М. К.* Проектирование электрических аппаратов. Л.: Энергоатомиздат, 1985. 448 с.

5. *Анго А.* Математика для электро- и радиоинженеров. М.: Наука, 1967. 780 с.

6. *Андерсон Дж.-А.* Дискретная математика и комбинаторика. М.: Вильямс, 2004. 960 с.

7. *Андреева Г.М.* Социальная психология: векторы новой парадигмы. Психологические исследования. 2009. № 1(3) [Электронный ресурс]: <http://www.psystudy.ru/index.php/num/2009n1-3/55-andreeva3>.

8. *Анищенко В. С., Нейман А. Б., Мосс Ф., Шиманский-Гейер Л.* Стохастический резонанс как индуцированный шумом эффект увеличения степени порядка // УФН. 1999. Т.169, № 1. С. 7–38.

9. *Антонов В. Ф.* Липидные поры: стабильность и проницаемость мембран // Соросовский образовательный журнал. 1998. № 10. С. 10–17.

10. *Антонов В. Ф., Смирнова Е. Ю., Шевченко Е. В.* Липидные мембраны при фазовых превращениях. М.: Наука, 1992. 125 с.

11. *Бахвалов Н. С., Воеводин В. В.* Современные проблемы вычислительной математики и математического моделирования. Т. 1. Вычислительная математика. М.: Наука, 2005. 343 с.

12. *Бегунов Б. Н.* Геометрическая оптика. М.: Изд-во МГУ, 1966. 209 с.

13. Большой энциклопедический словарь. М.: Астрель, 2008. 1248 с.

14. *Бондаренко И. Б., Гатчин Ю. А., Гераничев В. Н.* Синтез оптимальных искусственных нейронных сетей с помощью модифицированного генетического алгоритма // Научно-технический вестник информационных технологий, механики и оптики. 2012. № 2 (78). С. 51–55.

15. *Бугатенко Е. И., Титов В. С., Труфанов М. И.* Способ определения коэффициентов сферической аберрации. Патент № 2295712 РФ, МКИ G01M11/02. Опубл. 20.03.2007. Бюл. № 8. 6 с.

16. *Василенко О. Н.* Теоретико-числовые алгоритмы в криптографии. М.: МЦНМО, 2003. 328 с.

17. *Ватолин Д., Ратушняк А., Смирнов М., Юкин В.* Методы сжатия данных. Устройство архиваторов, сжатие изображений и видео. М.: ДИАЛОГ-МИФИ, 2003. 384 с.

18. *Висвани В.* Полный справочник по MySQL. М.: Вильямс, 2006. 528 с.

19. *Владимирский Б. М.* Активные процессы на Солнце и биосфера: Автореф. дис. д-ра физ.-мат. наук. Пущино, 1997. 28 с.

20. *Ву М., Девис Т., Нейдер Дж., Шрайнер Д.* OpenGL руководство по программированию. Библиотека программиста. СПб: Питер, 2006. 624 с.

21. *Вуколов Э. А.* Основы статистического анализа. Практикум по статистическим методам и исследованию операций с использованием пакетов STATISTICA и EXCEL. М.: Форум, 2004. 464 с.

22. *Выгодский М. Я.* Справочник по математике. М.: АСТ, 2011. 1055 с.

23. *Вяткин А. П.* Психология экономической социализации личности в условиях изменяющегося общества: Автореф. дис. д-ра педаг. наук. СПб: РГПУ им. А.И. Герцена, 2012. 45 с.

24. *Гаазе-Рапопорт М. Г., Кокшайский Н. В., Берг А. И., Брайнес С. Н.* и др. Проблемы бионики. М.: Наука, 1973. 528 с.

25. *Гад С. Я., Крючков А. Н., Яшин А. А.* Биофизика полей и излучений и биоинформатика / Под ред. *Е. И. Нефедова, А. А. Хадарцева, А. А. Яшина*. Тула: Тул ГУ, 2000. 286 с.

26. *Галушкин А. И.* Нейрокомпьютеры и их применение. М.: ИПРЖР, 2000. 528 с.

27. *Гельман Р. Н., Дунц А. Л.* Лабораторная калибровка цифровых камер с большой дисторсией // Геодезия и картография. 2002. № 7. С. 23–31.

28. *Гербер Р., Бик А., Смит К., Тиан К.* Оптимизация ПО. Сборник рецептов. СПб: Питер, 2010. 325 с.

29. *Гинзбург В. Л., Рухадзе А. А.* Волны в магнитоактивной плазме. М.: Наука, 1975. 256 с.

30. *Голыгин В. А., Сажин В. И., Унучков В. Е.* Коррекция модели ионосферы по данным о максимально-применимых частотах реперных радиолиний // Исследовано в России. 2006 [Электронный ресурс]: <http://zhurnal.ape.relarn.ru/articles/2006/255.pdf>.

31. *Гольденберг Л. Г., Матюшкин Б. Д., Поляк М. Н.* Цифровая обработка сигналов: Справочник. М.: Радио и связь, 1985. 312 с.

32. *Гороховатский В. А., Передрий Е. О.* Корреляционные методы распознавания изображений путём голосования систем фрагментов // Радіоелектроніка. Інформатика. Управління. 2009. № 1. С. 74–81.

33. *Гридин В. Н., Титов В. С., Труфанов М. И.* Адаптивные системы технического зрения. М.: Наука, 2009. 441 с.

34. *Гришенцев А. Ю.* Аппаратно-программный комплекс ИПЧ блок-схема реализации и некоторые результаты применения // Естественные и технические науки. М.: Компания Спутник +, 2008. № 4(36). С. 281–284.

35. *Гришенцев А. Ю.* Моделирование распределения плотности тока в сложном неоднородном проводнике. Ч. 1 // Науч.-техн. вестн. СПбГУИТМО. 2006. Вып. 29. С. 87–94.

36. *Гришенцев А. Ю.* Моделирование распределения плотности тока в сложном неоднородном проводнике. Ч. 2 // Науч.-техн. вестн. СПбГУИТМО. 2006. Вып. 29. С. 95–99.

37. *Гришенцев А. Ю.* Свойства преобразования *n*-мерных цифровых сигналов по базису прямоугольных всплесков // Науч.-техн. вестн. информационных технологий, механики и оптики. 2012. № 5 (81). С. 75–79.

38. *Гришенцев А. Ю.* Способ сжатия изображения. Патент № 2500067 РФ. Опубл. 27.11.2013.

39. *Гришенцев А. Ю.* Эффективное сжатие изображений на базе дифференциального анализа // Журн. радиоэлектроники. 2012. № 11 [Электронный ресурс]: <http://jre.cplire.ru/jre/nov12/index.html>.

40. *Гришенцев А. Ю., Бондаренко И. Б., Коробейников А. Г.* Программа анализа алгоритмов декомпозиции многомерных цифровых сигналов «Декомпозиция сигналов». Программа для ЭВМ № 2013614439. Опубл. 07.05.2013.

41. *Гришенцев А. Ю., Коробейников А. Г.* Декомпозиция *n*-мерных цифровых сигналов по базису прямоугольных всплесков // Науч.-техн. вестн. информационных технологий, механики и оптики. 2012. № 4 (80). С. 75–79.

42. *Гришенцев А. Ю., Коробейников А. Г.* Математическое моделирование процессов вертикального зондирования ионосферы // Тр. Междунар. конгр. по интеллектуальным системам и информационным технологиям AIS-IT'11. М.: Физматлит, 2011. Т.1. С. 418–424.

43. *Гришенцев А. Ю., Коробейников А. Г.* Обратная задача радиочастотного зондирования ионосферы Журн. радиоэлектроники. 2010. № 10 [Электронный ресурс]: <http://jre.cplire.ru/jre/oct10/6/text.html>.

44. *Гришенцев А. Ю., Коробейников А. Г.* Обратная задача радиочастотного зондирования ионосферы // Сб. тр. IV Всерос. конф. «Радиолокация и радиосвязь». М.: ИРЭ РАН, 2010. С. 201–207.

45. *Гришенцев А. Ю., Коробейников А. Г.* Программа обработки и анализа данных ионосферного спектрографа АИС-М «SkySpectrum». Программа для ЭВМ № 201161756928. Опубл. 28.09.2011.

46. *Гришенцев А.Ю., Коробейников А.Г.* Разработка модели распределения плотности токов при возбуждении ионосферы высокочастотным облучением // Изв. вузов. Приборостроение. 2010. №12 (53). С. 41–47.

47. *Гришенцев А. Ю., Коробейников А. Г.* Разработка математической модели решения обратной задачи вертикального зондирования ионосферы на основе методов распознавания образов // Сб. матер. Всерос. науч.-практ. конф. «Информационные технологии в профессиональной деятельности и научной работе». Йошкар-Ола: Марийский государственный технический университет, 2011. Ч. 1. С. 46–53.

48. *Гришенцев А. Ю., Коробейников А. Г.* Разработка модели решения обратной задачи вертикального зондирования ионосферы // Науч.-техн. вестн. информационных технологий, механики и оптики. 2011. № 2 (72). С. 109–112.

49. *Гришенцев А. Ю., Коробейников А. Г.* Способ построения спектра *n*-мерных неразделимых цифровых сигналов. Патент № 2484523 РФ. Заявл. от 29.06.2011.

50. *Гришенцев А. Ю., Коробейников А. Г.* Увеличение скорости сходимости метода конечных разностей на основе использования промежуточного решения // Сб. матер. Всерос. науч.-практ. конф. «Информационные технологии в профессиональной деятельности и научной работе». Йошкар-Ола: Марийский государственный технический университет, 2012. Ч. 2. С. 9–14.

51. *Гришенцев А. Ю., Коробейников А. Г.* Улучшение сходимости метода конечных разностей с помощью вычисления промежуточного решения // Науч.-техн. вестн. информационных технологий, механики и оптики. 2012. № 3 (79). С. 124–127.

52. *Гришенцев А. Ю., Муромцев Д. И.* Система управления данными наблюдений солнечно-земной физики «МІ». Программа для ЭВМ № 2011615714. 21.07.2011.

53. *Гришенцев А. Ю., Ярош А. М.* Анализ зависимости числа клинических случаев от некоторых факторов природного происхождения // Сб. тр. «Информатика и вычислительная техника» / Под ред. *В. Н. Негоды*. Ульяновск: УлГТУ, 2010. С. 579–586.

54. *Гришенцев А. Ю., Ярош А. М., Коробейников А. Г.* Особенности влияния солнечного радиоизлучения на число клинических состояний человека // Журн. радиоэлектроники. 2010. № 11 [Электронный ресурс]: <http://jre.cplire.ru/jre/nov10/3/text.html>.

55. *Гусятинский И. А., Немировский А. С., Соколов А. В., Троицкий В. Н.* Дальняя тропосферная радиосвязь. М.: Связь, 1968. 248 с.

56. *Давыдов А. В.* Цифровая обработка сигналов: Тематические лекции. Екатеринбург: УГГУ, 2007 [Электронный ресурс]: <http://www.prodav.narod.ru/dsp/index.html>.

57. *Девятков Н. Д., Голант М. Б., Бецкий О. В.* Миллиметровые волны и их роль в процессах жизнедеятельности. М.: Радио и связь, 1991. 168 с.

58. *Девятков Н. Д., Голант М. Б., Бецкий О. В.* Особенности медико-биологического применения миллиметровых волн. М.: ИРЭ РАН, 1994. 164 с.

59. *Дейт К. Дж.* Введение в системы баз данных. М.: Вильямс, 2005. 1328 с.

60. *Демирчан К. С., Нейман Л. Р., Коровкин Н. В., Чечурин В. Л.* Теоретические основы электротехники. СПб: Питер, 2006. Т. 2. 576 с.

61. *Домбровский А. Н.* Стохастический резонанс и фильтрация сигналов в нелинейных радиотехнических системах: Автореф. дис. канд. техн. наук. М., 2009.

62. *Думин Ю. В.* Концентрация носителей заряда в метастабильной плазме с сильной кулоновской неидеальностью // Прикладная физика. 1999. № 5. С. 18–21.

63. *Зверев В. А., Стромков А. А.* Выделение сигналов из помех численными методами. Нижний Новгород: ИПФ РАН, 2001. 188 с.

64. *Зевеке Г. В., Ионкин П. А., Нетушил А. В.* Основы теории цепей. М.: Энергия, 1975. 752 с.

65. *Зимичев А.М.* Психология политической борьбы. СПб: Санта, 1993. 160 с.

66. *Зуев В. В., Розова С. С.* Проблема способа бытия таксона в биологической таксономии // Вопросы философии. 2003. № 2. С. 90–103.

67. *Ильин В. А., Поздняк Э. Г.* Линейная алгебра. М.: Наука, 1974. 296 с.

68. *Ильичёв А. Т.* Уединённые волны в моделях гидродинамики. М.: Физматлит, 2003. 256 с.

69. *Калантаров П. Л., Цейтлин Л. А.* Расчет индуктивностей. Справочная книга. Л.: Энергоатомиздат, 1986. 488 с.

70. *Калинин А. И., Черенкова Е. Л.* Распространение радиоволн и работа радиолиний. М.: Связь, 1971. 450 с.

71. *Кант И.* Критика чистого разума / Пер. с нем. *Н. Лосского*; под ред. *Ц. Г. Арзакаяна и М. И. Иткина.* М.: Эксмо, 2006. 736 с.

72. *Кантор Г.* Труды по теории множеств. М.: Наука, 1985. 431 с.

73. *Касперски К.* Техника оптимизации программ (+CD). СПб: БХВ-Петербург, 2003. 464 с.

74. *Кестр У.* Проектирование систем цифровой и смешанной обработки сигналов. М.: Техносфера, 2010. 328 с.

75. *Ким Ю. В.* Модернизация ионосферной станции «АИС». Отчёт по теме № 690. Троицк: ИЗМИРАН, 2002. 24 с.

76. *Киреев О. С.* Нейрокалибровка стереопары // Математичні машини і системи. 2005. № 1. С. 13–25.

77. *Клюев Н. И.* Информационные основы передачи сообщений. М.: Сов. радио, 1966. 360 с.

78. *Корис Р., Шмидт-Вальтер Х.* Справочник инженера-схемотехника. М.: Техносфера, 2008. 608 с.

79. *Коробейников А. Г., Гатчин Ю. А.* Математические основы криптологии. СПб: СПбГУ ИТМО, 2004. 106 с.

80. *Коробейников А. Г., Копытенко Ю. А., Исмагилов В. С, Гришенцев А. Ю.* Интеллектуальные системы магнитных измерений на железнодорожных сортировочных станциях // Сб. докл. междунар. науч.-практ. конф. «Автоматизация и механизация технологических процессов на сортировочных станциях». М., 2010. С. 73–75.

81. *Коротков К. Г.* Основы ГРВ биоэлектрографии. СПб: СПбГИТМО, 2001. 360 с.

82. *Котельников В. А.* О пропускной способности «эфира» и проволоки в электросвязи // Сб. Всесоюз. энергетический комитет. Матер. к I съезду по вопросам и технической реконструкции дела связи и развития слаботочной промышленности. М.: Управление связи РККА, 1933. С. 1–19.

83. *Котельникова Н. В.* К 100-летию со дня рождения академика Котельникова Владимира Александровича. 2008 [Электронный ресурс]: <http://www.ras.ru/news/shownews.aspx?id=59dc9c27-d249-486e-b537-2edfb02a0ede>.

84. *Кувшинов С. С.* Методы сокрытия больших объёмов данных на основе стеганографии: Дис. канд. техн. наук. СПб: СПбГУ ИТМО, 2010. 116 с.

85. *Лайонс Р.* Цифровая обработка сигналов. М.: Бином-Пресс, 2009. 656 с.

86. *Мандельштам Л. И.* Полное собрание трудов. М.: Изд-во Академии наук СССР, 1955. 512 с.

87. Мировые центры данных (МЦД) России и Украины [Электронный ресурс]: <http://www.wdcb.ru>.

88. *Михайлов А. П., Чибуничев А. Г., Курков В. М., Piatti E. J.* Применение цифровых неметрических камер и лазерных сканеров для решения задач фотограмметрии. Ракурс. 2003 [Электронный ресурс]: <http://www.racurs.ru/www_download/articles/Camaras_digitales_rus.pdf>.

89. *Мукушев Б. А.* Проблемы формирования нелинейного стиля мышления личности // Alma mater. М.: РУДН, 2009. С. 16–22.

90. *Намгаладзе А. А., Юрик Р. Ю.* Математическое моделирование возмущений верхней атмосферы Земли. Российский фонд фундаментальных исследований [Электронный ресурс]: <http://www.rfbr.ru/pics/28326ref/file.pdf>.

91. *Нефедов Е. И., Протопопов А. А., Хадарцев А. А., Яшин А. А.*, Биофизика полей и излучений и биоинформатика. Ч. 1. Физико-биологические основы информационных процессов в живом веществе. Тула: Изд-во ТулГУ, 1998. 333 с.

92. *Нечаев Д. А., Гришенцев А. Ю.* Исследование работы прибора «ИПЧ» при различных значениях влажности // Изв. вузов. Приборостроение. 2006. № 2(49). С. 26–30.

93. *Новгородцев А. Б.* Теория электромагнитного поля. СПб: Изд-во СПбГТУ, 1994.

94. *Новиков С. П., Тайманов И. А.* Современные геометрические структуры и поля. М.: МЦНМО, 2005. 584 с.

95. *Оппенгейм А., Шафер Р.* Цифровая обработка сигналов. М.: Техносфера, 2009. 856 с.

96. *Панасенко С. П.* Алгоритмы шифрования. СПб: БХВ-Петербург, 2009. 576 с.

97. *Пискунов Н. С.* Дифференциальное и интегральное исчисления. М.: Интеграл-Пресс, 2005. Т. 2. 544 с.

98. *Покровский В. М., Коротько Г. Ф.* Физиология человека. М.: Медицина, 1997. Т. 1. 448 с.

99. *Пономаренко Г. Н.* Физиотерапия в косметологии. СПб: ВМедА, 2002. 356 с.

100. *Прата С.* Язык программирования C++. М.: Вильямс, 2013. 1248 с.

101. Программа моделирования электромагнитных, тепловых и механических задач, ELCUT. ООО «Тор», 2012 [Электронный ресурс]: <http://elcut.ru/>, MicroCap. MicroCad, Spectrum software [Электронный ресурс]: <http://www.spectrum-soft.com/>.

102. *Рубин А. Б.* Биофизика. М.: Наука, 2004. Т. 2. 448 с.

103. *Самарский А. А., Вабищевич П. Н.* Численные методы решения обратных задач математической физики. М.: Изд-во ЛКИ, 2009. 480 с.

104. *Самойлов В. О.* Медицинская биофизика. СПб: СпецЛит, 2007. 560 с.

105. *Селмон Д.* Сжатие данных, изображений и звука. М.: Техносфера, 2006. 368 с.

106. *Семёнов А. М., Сикарев А. А.* Широкополосная радиосвязь. М.: Воениздат, 1970. 280 с.

107. *Сергеенко В. С., Баринов В. В.* Сжатие данных, речи, звука и изображений в телекоммуникационных системах. М.: РадиоСофт, 2011. 360 с.

108. *Серов А. В.* Эфирное цифровое телевидение DVB-T/H/. СПб: БХВ-Петербург, 2010. 464 с.

109. *Седжевик Р.* Алгоритмы на C++. М.: Вильямс, 2011. 1056 с.

110. *Скиена С.* Алгоритмы. Руководство по разработке. СПб: БХВ-Петербург, 2011. 720 с.

111. *Смирнов В. М., Смирнова Е. В.* Реконструкция пространственно-временной структуры ионосферы по данным спутниковых наблюдений // Современные проблемы дистанционного зондирования Земли из космоса». М.: Изд-во «Институт космических исследований РАН» , 2008. Т. 5. С. 561–566.

112. *Смит С.* Цифровая обработка сигналов. М.: Додека-XXI, 2011. 720 с.

113. Создание комплекса научной аппаратуры с новыми информационными каналами регистрации корпускулярного и электромагнитного излучений Солнца; приоритетные результаты в наблюдениях солнечной активности и ее воздействий на Землю со спутника КОРОНАС-Ф (2001–2005 гг.). ИЗМИРАН, 2007 [Электронный ресурс]: <http://www.izmiran.rssi.ru/achievments/press-release>.

114. *Солонина А. И., Улахович Д. А., Арбузов С. М., Соловьёва Е. Б.* Основы цифровой обработки сигналов. СПб: БХВ-Петербург, 2005. 768 с.

115. *Стивенс Р., Раго С.* UNIX. Профессиональное программирование. СПб: Символ-Плюс, 2007. 1040 с.

116. *Стрелкова А. Н., Титов В. С., Труфанов М. И.* Патент № 2292023 РФ, МКИ G01M11/02. Способ определения комы оптической системы. Опубл. 20.01.2007. Бюл. № 2. 5 с.

117. *Фадеев М. А., Чупрунов Е. В.* Лекции по атомной физике. М.: Физматлит, 2008. 612 с.

118. *Формалёв В. Ф., Ревезников Д. Л.* Численные методы. М.: Физматлит, 2004. 400 с.

119. *Харкевич А. А.* Передача сигналов модулированных шумом // Электросвязь. 1957. № 11. С. 42–46.

120. *Харт Дж. М.* Системное программирование в среде Windows. М.: Вильямс, 2005. 592 с.

121. *Хлыбов Е. С., Гаврилов Б. Г., Егоров Д. В.* Исследование влияния сильных сейсмических событий на полное электронное содержание в ионосфере // Тр. Междунар. Байкальской молодежной науч. шк. по фундаментальной физике. 2006. С. 177–180.

122. *Хортон А.* Visual C++ 2010. М.: Диалектика, 2010. 1216 с.

123. *Чобану М.* Многомерные многоскоростные системы обработки сигналов. М.: Техносфера, 2009. 480 с.

124. *Шапиро Л., Стокман Дж.* Компьютерное зрение. М.: Бином, 2006. 752 с.

125. *Шовенгердт Р. А.* Дистанционное зондирование. Модели и методы обработки изображений. М.: Техносфера, 2010. 560 с.

126. *Штарк Г.-Г.* Применение вейвлетов для ЦОС. М.: Техносфера, 2007. 192 с.

127. *Эхтер Ш., Робертс Дж.* Многоядерное программирование. СПб: Питер, 2010. 316 с.

128. *Яковлев О. И., Павельев А. Г., Матюгов С. С.* Спутниковый мониторинг Земли: Радиозатменный мониторинг атмосферы и ионосферы. М.: Либроком, 2010. 208 с.

129. Яковлев О. И., Якубов В. П., Урядов В. П., Павельев А. Г. Распространение радиоволн. М.: Ленанд, 2009. 496 с.

130. A Big Surprise from the Edge of the Solar System. NASA. 2012 [Electronic resource]: <http://www.nasa.gov/mission_pages/voyager/ heliosphere-surprise.html>.

131. Background to Ionospheric Sounding [Electronic resource]: <http://ulcar.uml.edu/DPS.htm>.

132. *Benzi R., Sutera A., Vulpiani A.* Some Problems in Statistical Physics // J. Phys / Ed. by *G. H. Weiss*. SIAM, Philadelphia, 1992. P. 453–467.

133. *Benzi R., Parisi G., Sutera A., Vulpiani A.* Stochastic resonance in climatic change // Tellus. 1982. Vol. 34. P. 10–16.

134. *Biggs D.S.C.* Acceleration of Iterative Image Restoration Algorithms // Applied Optics. 1997. Vol. 36, N 8. P. 1766–1775.

135. *Boschi D., Branca S., DeMartini F., Hardy L., Popescu S.* Experimental Realization of Teleporting an Unknown Pure Quantum State via Dual Classical and Einstein-Podolsky-Rosen Channels // Phys. Rev. Lett. 1998. Vol. 80. P.1121–1125.

136. *Bouwmeester D., Pan J.-W., Mattle K., Eibl M., Weinfurter H., Zeilinger A.* Experimental Quantum Teleportation // Nature. 1997. Vol. 390. P. 575–579.

137. *Bremermann H. J.* Optimization through Evolution and Recombination // Self-Organizing Systems / Ed. by *M. C. Yovits and S. Cameron*. Washington: Spartan, 1962. P. 93–106.

138. Canadian Advanced Digital Ionosonde. System Manuals / Scientific Instrumentation Limited. Canada, 2009. 22 p.

139. *Cooley J. W., Tukey J. W.* An Algorithm for the Machine Calculation of Complex Fourier Series // Mathematics Computation. 1965. Vol. 19. P. 297–301.

140. *DeHaan G.* Progress in Motion Estimation for Video Format Conversion// IEEE Trans. on Consumer Electron. 2000. Vol. 46, N 3. P. 449–450.

141. *DeMarco G.* Minority Rules: Scientists Discover Tipping Point for the Spread of Ideas. Rensselaer Polytechnic Institute (RPI). 2011 [Electronic resource]: <http://news.rpi.edu/update.do?artcenterkey=2902>.

142. *Douglass J. K. et al.* Noise enhancement of information transfer in crayfish mechanoreceptors by stochastic resonance // Nature. 1993. Vol. 365. P. 337–340.

143. *Einstein A., Podolsky B., Rosen N.* Can Quantum-Mechanical Description of Physical Reality Be Considered Complete? // Phys. Rev. 1935. Vol. 47(10). P. 777–780.

144. *Fauve S., Heslot F.* Stochastic resonance in a bistable system // Phys. Lett. A. 1983. Vol. 97. P. 5–7.

145. *Fraenkel A, Klein S.* Robust Universal Complete Codes for Transmission and Compression // Discrete Applied Mathematics. 1996. Vol. 64. P. 31–55.

146. *Gammaitoni L et al.* Observation of stochastic resonance in bistable electron-paramagnetic-resonance systems // Phys. Rev. Lett. 1991. Vol. 67. P. 1799–1802.

147. *Gray J.* The Transaction Concept: Virtues and Limitations. Tandem Computers Incorporated, 1981.

148. *Grigorenko A. N., Nikitin P. I., Konov V. I.* Magnetostochastic Resonance // Sov. Phys. JETP Lett. 1990. Vol. 52. P. 993.

149. Guide to Reference and Standard Ionosphere Models. American Institute of Aeronautics and Astronautics, 1999. 55 p.

150. *Hanisch R. J., White R. L., and Gilliland R. L.* Deconvolution of Hubble Space Telescope Images and Spectra. Deconvolution of Images and Spectra / Ed. by *P. A. Jansson.* Boston, MA: Academic Press, 1997. P. 310–356.

151. *Heikkila J., Silven O. A.* Four-step Camera Calibration Procedure with Implicit Image Correction // CVPR97. 1997. P. 1106–1112.

152. *Heisenberg W.* Physikalische Prinzipien der Quantentheorie. Mannheim, 1958. 45 s.

153. ITU-R Recommendation BT.601-7. Studio encoding parameters of digital television for standard 4:3 and wide-screen 16:9 aspect ratios. ITU, Geneva, Switzerland, 2011.

154. ITU-R Recommendation BT.709, Basic Parameter Values for the HDTV Standard for the Studio and International Programme Exchange. ITU, Geneva, Switzerland, 2002.

155. *Keith J.* Video Demystified: a Handbook for the Digital Engineer. LLH Technology Publishing, 2001.

156. *Levin J. E., Miller J. P.* Broadband Neural Encoding in the Cricket Cereal Sensory System Enhanced by Stochastic Resonance // Nature. 1996. Vol. 380. P. 165–168.

157. *Longuet-Higgins M. S.* Capillary-gravity Waves of Solitary Type and Envelope Solitons on Deep Water // J. Fluid Mech. 1993. Vol. 252. P. 703–711.

158. *Malvar H. S., Sullivan G. J.* Transform, Scaling & Color Space Impact of Professional Extensions. ISO/IEC JTC/SC29/WG11 and ITU-T SG16 Q.6 Document JVT-H031. Geneva, 2003.

159. MATLAB GPU Computing Support for NVIDIA CUDA-Enabled GPUs. 2012 [Electronic resource]: <http://www.mathworks.com/discovery/matlab-gpu.html>.

160. *McNamara B., Wiesenfeld K., Roy R.* Observation of Stochastic Resonance in a Ring Laser // Phys. Rev. Lett. 1988. Vol. 60. P. 2626–2629.

161. MSDN. Microsoft, 2012 [Electronic resource]: <http://msdn.microsoft.com>.

162. National Oceanic and Atmospheric Administration (NOAA) [Electronic resource]: <http://www.noaa.gov/>.

163. NOAA's National Geophysical Data Center (NGDC) [Electronic resource]: <http://www.ngdc.noaa.gov/>.

164. *Rasmussen C. E., Schunk R. W.* A Three-dimensional Time-dependent Model of the Plasmasphere // J. Geophys. Res. 1990. Vol. 95. P. 6133–6144.

165. *Richmond A. D., Ridley E. C., Roble R. G.* A Thermosphere/ionosphere General Circulation Model with Coupled Electrodynamics // Geophys. Res. Lett. 1992. Vol. 19, N 6. P. 601–604.

166. *Rogers D.* Procedural Elements for Computer Graphics. McGraw-Hill, 1985. P. 68–81.

167. *Rycroft M. J.* Solar-terrestrial energy program: handbook of ionospheric models // J. of Atmospheric and Solar-Terrestrial Physics. Elsevier Science Publishing Company, Inc., 1998. P. 403–404.

168. *Pei X., Wilkens L., Moss F.* Noise-mediated Spike Timing Precision from Aperiodic Stimuli in an Array of Hodgekin-Huxley–type Neurons // Phys. Rev. Lett. 1996. Vol. 77. P. 4679–4682.

169. *Piggott W. R., K. Rawer.* URSI handbook of Ionogram Interpritation and Reduction. INAG (Ionospheric Network Advisory Group). World data center A. National Academy of Sciences. Washington, 1972. 145 p.

170. *Price D. J. de Solla.* An Ancient Greek Computer // Scientific American. 1959. June. P. 60–67.

171. *Saupe D., Hamzaoui R., Hartenstein H.* Fractal Image Compression – An introductory overview // Fractal Models for Image Synthesis, Compression, and Analysis. ACM SIGGRAPH'96 [Electronic resource]: <Course Notes, http://www.factbites.com/topics/Fractal-compression>.

172. *Shannon C. E.* A mathematical theory of communication // The Bell System Techn. J. 1948.Vol. 27. P. 379–423, 623–656.

173. *Simonotto E., Rani M., Seife C. et al.* Visual Perception of Stochastic Resonance // Phys. Rev. Lett. 1997. Vol. 78. P. 1186–1189.

174. Space Weather Prediction Center [Electronic resource]: <http://www.swpc.noaa.gov/>.

175. The Solar Dynamics Observatory. NASA, 2012 [Electronic resource]: <http://sdo.gsfc.nasa.gov/>.

176. *Tsai R. Y.* A Versatile Camera Calibration Technique for High-accuracy 3D Machine Vision Metrology Using off-theshelf TV Cameras and Lenses // IEEE Int. J. on Robotics and Automation. 1987. Vol. 3. P. 323–344.

177. United States Geological Survey's (USGS) [Electronic resource]: <http://earthquake.usgs.gov/>.

178. University of Southern California. The USC-SIPI Image Database. US, 2012 [Electronic resource]: <http://sipi.usc.edu/database/>.

Научное издание

Гришенцев Алексей Юрьевич
Коробейников Анатолий Григорьевич

МЕТОДЫ И МОДЕЛИ ЦИФРОВОЙ ОБРАБОТКИ ИЗОБРАЖЕНИЙ

Налоговая льгота – Общероссийский классификатор продукции
ОК 005-93, т. 2; 95 3004 – научная и производственная литература

Подписано в печать 02.06.2015. Формат 60×84/16. Печать цифровая.
Усл. печ. л. 12,0. Тираж 32. Заказ 13160b.

Отпечатано с готового оригинал-макета, предоставленного
Издательством Политехнического университета,
в Типографии Политехнического университета.
195251, Санкт-Петербург, Политехническая ул., 29.
Тел.: (812) 552-77-17; 550-40-14.

ISBN 978-5-7422-4892-7