Российская Академия Наук

Институт прикладной математики им. М.В. Келдыша

На правах рукописи

Переберин Антон Валерьевич

Многомасштабные методы синтеза и анализа изображений

Специальность 05.13.11 — математическое и программное обеспечение вычислительных машин, комплексов и компьютерных сетей

ДИССЕРТАЦИЯ

на соискание ученой степени

кандидата физико-математических наук

Научный руководитель —

кандидат физико-математических наук

Ю.М. Баяковский

Москва — 2002

Оглавление

Введение

Повышение эффективности обработки информации является актуальной задачей компьютерной графики. Требования к реалистичности генерируемых изображений постоянно растут. Это приводит увеличению объемов обрабатываемой информации, к усложнению алгоритмов обработки, и, как следствие, к росту вычислительных затрат. В то же время, для многих приложений (например, игровых) необходима очень высокая скорость обработки графической информации. Рост производительности оборудования решает эту проблему, как показывает практика, лишь отчасти.

Возможны следующие пути повышения эффективности обработки графической информации.

Первый путь — сокращение объемов данных. Описание объекта может содержать избыточную или несущественную информацию, которую можно отбросить. Допустим, имеется сеточное представление объекта — некоторого рельефного изображения на плоской поверхности (см. пример на с. 111). Чтобы передать сложную структуру изображения, сетка должна быть достаточно мелкой (и, следовательно, содержать большое число вершин и треугольников). Однако заметно, что выдерживать постоянный шаг сетки для всего объекта не требуется. Мелкая сетка явно избыточна для задания фрагмента плоскости. Некоторые участки изображения также можно

представить с помощью более крупных треугольников. Таким образом, объект можно представить сеткой с меньшим числом треугольников. Затраты на обработку такой сетки снижаются, также уменьшается расход памяти для ее хранения.

Существенность информации может определяться не только особенностями объекта, но и особенностями *отображения* объекта. Предположим, что сетку из вышеприведенного примера потребовалось отобразить в область малого размера. При этом может получиться, что реальный размер некоторых треугольников сетки составит менее 1-2 пикселов. Очевидно, что в таком случае имеет смысл использовать более крупную сетку.

Второй путь — поиск удобных для обработки представлений информации. Например, представлений, которые обеспечивают эффективную реализацию таких упомянутых выше операций, как сокращение общего объема данных и выбор информации, существенной для отображения объекта при заданных условиях. Также желательно, чтобы одно представление могло использоваться для решения сразу нескольких подзадач сложной многоэтапной обработки объекта. В этом случае отсутствие необходимости на каждом этапе преобразовывать объект в новое представление может упростить общую процедуру обработки и повысить эффективность ее реализации.

Третий путь — разработка более производительных алгоритмов обработки информации. Повышение производительности может достигаться разными методами, например, оптимизацией существующих алгоритмов, применением более эффективных численных методов, распараллеливанием, аппаратной поддержкой части вычислений и т.д. Очевидно, что разработ-

ка производительных алгоритмов тесно связана с поиском представлений информации, обеспечивающих эффективную обработку.

Допустим теперь, что существует описание объекта в виде многослойной структуры, обладающей следующим свойством: первый слой содержит информацию, достаточную для грубого (с низким разрешением) приближения объекта; при добавлении информации из каждого последующего слоя степень детализации постепенно увеличивается, пока объект не будет восстановлен полностью (то есть с максимальным разрешением). Такое представление обладает целым рядом полезных свойств. Прежде всего, оно позволяет выделить на изображении фрагменты, которые при переходе от слоя к слою меняются либо слабо, либо, напротив, сильно. Слабые изменения свидетельствуют о том, что данный фрагмент достаточно хорошо представим и с невысоким разрешением. Это позволяет исключить информацию, соответствующую такому фрагменту, из последующих слоев, и обеспечивает, таким образом, сжатие информации, т.е. сокращение объемов данных. Сильные изменения, наоборот, соответствуют фрагментам, которые требуется представлять с высоким разрешением, их локализация позволяет выделять контуры изображения, мелкие детали и т.д. Многослойное представление открывает широкие возможности и для редактирования объектов: вносить изменения можно не в объект целиком, а либо в одно из приближений объекта, либо в детализирующие слои. Иерархическая структура позволяет эффективно отображать объект в зависимости от конкретных условий (параметров графического устройства, расположения объекта в сцене, положения наблюдателя). Возможна демонстрация объекта с разрешением, возрастающим по мере получения детализирую-

щей информации. Такая операция (т.н. прогрессивная передача), полезна в случае, когда невозможно моментально получить полностью описание объекта, например, при передаче по сети, или если нужно получить изображения большого числа объектов, не требуя их воспроизведения с высоким разрешением, например, при просмотре результатов запросов к базам данных с графической информацией.

Таким образом, иерархическое представление позволяет обеспечить разностороннюю обработку объекта, включая сжатие данных и управление уровнем детализации. Кроме того, как будет показано ниже, построение и обработка таких представлений может осуществляться с помощью достаточно простых и эффективных алгоритмов.

Иерархическое представление, обладающее подобными свойствами, в литературе называют *многомасштабным*. Примером конструкций, обеспечивающих многомасштабное представление информации, служат *пирамиды лапласианов* и *гауссианов*, предложенные в [24]. Идеи, использованные при построении этих пирамид, легли в основу теории *вейвлет-анализа* (или *анализа всплесков*)[2, 3, 11, 15, 25, 29, 45, 56, 73, 77] — инструмента, который активно используется в настоящее время для работы с многомасштабными представлениями данных самой различной структуры.

Вейвлет-анализ — это разложение сигнала по специальному базису. Базисные функции (*вейвлеты*) получаются сдвигом и масштабированием (сжатием или растяжением) одной функции — *порождающего вейвлета*. Как правило, вейвлетом является функция с компактным носителем, или функция, быстро убывающая на бесконечности, среднее значение которой равно нулю.

Масштаб в вейвлет-анализе является в определенном смысле аналогом частоты в анализе Фурье. Однако в отличие от анализа Фурье каждому значению масштаба вейвлет-анализа соответствует вообще говоря бесконечное количество сдвинутых друг относительно друга локализованных в пространстве функций. Таким образом, вейвлет-анализ является частотно-пространственным, что принципиально отличает его от строго частотного анализа Фурье.

Другим отличием является то, что в случае вейвлет-анализа в наиболее общей постановке не конкретизируется не только сам порождающий вейвлет, но и то, какие именно его копии участвуют в разложении. Отсюда очевидно, что термин «вейвлет-преобразование» является обозначением целого класса разложений. Анализ Фурье, как известно, является разложением по фиксированной системе функций.

Считается, что начало вейвлет-анализу было положено в работах А. Хаара еще в начале XX века [40]. В дальнейшем предпринимались попытки создания иерархических базисов для решения различных задач, но они не были объединены единой теорией и, следовательно, не имели единого подхода.

В конце 80-х годов С. Малла вводит понятие *многомасштабного анализа* [55], и определяет общий подход для поиска различных вейвлет-базисов. Им же разрабатывается основной алгоритм вычисления вейвлет-преобразований для дискретных сигналов, что открывает широкие возможности для практической реализации метода. С этого момента теория и практика вейвлет-анализа начинают активно развиваться. В 1992 году появляется классическая работа И. Добеши «Десять лекций по вейвле-

там» [29] (в 2001 году издана на русском языке [3]). Эта книга, а также некоторые другие издания (напр., [77]), посвящены строгому изложению теории вейвлет-анализа. Выходит в свет несколько книг, в которых содержится подробное введение в вейвлет-анализ, а также рассматривается широкий круг прикладных задач [25, 26, 56]. В середине 90-х годов В. Свелденс предлагает новый метод вычисления вейвлет-преобразований, названный *лифтингом* [30, 74], который несколько более эффективен, чем классический алгоритм, предложенный Малла, и, кроме того, позволяет расширить класс вейвлет-преобразований.

Являясь достаточно эффективным инструментом обработки сигналов различной природы, вейвлет-анализ находит применение в математике, физике, астрономии, медицине, радиоэлектронике и других отраслях.

Наиболее распространенным примером применения вейвлет анализа в компьютерной графике и обработке изображений является сжатие изображений [15, 31, 57, 66, 68]. Во многих публикациях можно встретить упоминание о том, что один из первых алгоритмов был разработан для сжатия изображений отпечатков пальцев по заказу ФБР США, где и сейчас успешно используется. Разработчики стандарта JPEG-2000 утверждают, что их алгоритм сжатия основан на вейвлет-преобразовании.

Кроме этого вейвлеты применяются для обработки практически всех основных графических объектов: кривых [25, 34], поверхностей [22, 32, 39, 53, 54], сплошных трехмерных тел [42]. Отдельно можно отметить применение вейвлет-анализа в таких сложных задачах графики, как моделирование освещенности методом излучательности [38].

Вейвлет-анализ находит применение и в задачах компьютерного зрения,

распознавание и классификации образов [44, 76].

В 1996 году выходит книга Дж. Столнитза и др. «Вейвлеты в компьютерной графике. Теория и приложения» [73], в которой, кроме необходимого введения в теорию, описываются наиболее характерные приложения вейвлет-анализа в графике, а также содержится большой библиографический список.

Вейвлет-анализ является не единственной альтернативой анализу Фурье. Примерами могут служить разложения по базисам Габора [36] и Эрмита [58, 59]. Базис Габора фактически является вейвлет-базисом, однако не существует многомасштабного анализа для такого базиса и, как следствие, для вычисления разложения по этому базису не применимы быстрые алгоритмы вейвлет-преобразований. Базис Эрмита является ортонормированным, его структура близка к тригонометрическому базису (каждому уровню разрешения или частоте соответствует только одна базисная функция), однако базисные функции имеют пространственную локализацию. Оба базиса применяются в различных задачах обработки сигналов.

В настоящее время вейвлет-анализу уделяется все больше внимания и в отечественных исследованиях, однако, пока этот аппарат еще не получил широкого распространения. Возможно, это связано с недостатком литературы по основам вейвлет-анализа, она выходит небольшими тиражами и не всегда доступна [11, 15]. Переводы трудов зарубежных авторов стали появляться совсем недавно [3, 20].

Тем не менее, появляются работы как теоретического плана [10, 51, 78], так и посвященные различным прикладным задачам [4, 9, 18] (см. также списки публикаций отечественных авторов в [3, 20]).

Что касается компьютерной графики и обработки изображений, то отечественных работ в этой области весьма мало, и посвящены они, в основном, сжатию изображений [7, 16]. Из других задач можно упомянуть работу об использовании вейвлетов для решения уравнения излучательности [8].

Кроме того, следует отметить работы по обработке изображений с помощью базиса Эрмита [48], а также оригинальный алгоритм сжатия изображений, который основан на иерархическом, но не вейвлетном представлении данных [43].

Цель работы

Разработка многомасштабных методов анализа и синтеза графических объектов разной структуры. Изучение возможностей адаптации этих методов и реализующих их алгоритмов к особенностям конкретных задач и требованиям приложений. Осуществление с помощью указанных методов процессов многоэтапной обработки графической информации.

Научная новизна работы

В работе рассмотрены новые способы решения нескольких задач компьютерной графики, основанные на многомасштабном представлении информации. Предложен метод адаптивной триангуляции на основе дерева вейвлет-преобразований и его модификация для решения задачи расчета и представления освещенности. Предложено применение B-сплайнового вейвлет-преобразования для обработки и отображения линий уровня. Предложена модель описания стохастических текстур, близкая по структуре к разложению сигнала по вейвлет-базису.

Новым является комплексный подход к применению многомасштабных методов для задач, требующих многоэтапной обработки информации. Предлагается использовать одно и то же представление для решения возможно большего числа подзадач. Такой подход упрощает общую процедуру обработки и повышает эффективность ее реализации. Кроме того, становится возможным расширять функциональные возможности метода, внося в него минимум дополнений.

Дополнительно можно отметить, что в процессе разработки модели текстур была сделана заявка на новое, «функциональное» расширение вейвлет-преобразования. (Однако изучение свойств, возможностей, способов реализации и класса приложений такого расширения является предметом самостоятельного исследования).

Практическая значимость

Разработаны и доведены до реализации методы решения нескольких актуальных задач компьютерной графики. Реализованные алгоритмы удовлетворяют требованиям и ограничениям, которые были сформулированы при постановке задач. В частности, алгоритм генерации и нанесения текстур разрабатывался с учетом возможной аппаратной реализации в графических ускорителях. Метод построения изолиний внедрен в программный продукт, разработанный для геологических расчетов.

Апробация работы и публикации

Основные результаты диссертации докладывались на конференциях по компьютерной графике и машинному зрению «ГрафиКон'97», «Графи-

Кон'99», «ГрафиКон'2000», семинаре по компьютерной графике и обработке изображений Ю.М. Баяковского (ф-т ВМиК МГУ), объединенном семинаре ИПМ им. М.В. Келдыша РАН и МГТУ им. Н.Э. Баумана и опубликованы в работах [13, 14, 62, 63, 64].

Структура диссертации

Диссертация состоит из четырех глав и приложения.

Первая глава содержит краткое описание аппарата вейвлет-преобразований одно- и двумерных дискретных сигналов, который применяется при решении задач, рассматриваемых в последующих главах.

Во второй главе описывается алгоритм построения адаптивных треугольных сеток для представления графических объектов, параметризуемых на плоскости. Алгоритм основан на древовидной структуре многомасштабного анализа информации. Рассматривается модификация этого алгоритма для приложения — реконструкции функции освещенности по данным, полученным с помощью прямой трассировки лучей методом Монте-Карло.

В третьей главе обсуждается применение вейвлет-анализа для решения задачи построения линий уровня (изолиний) на плоскости, рассматриваются вопросы сглаживания линий, масштабирования и вывода на графическое устройство с заданными характеристиками.

В четвертой главе рассматривается многомасштабная модель для описания некоторого класса стохастических текстур. С помощью этой модели возможно создание как реалистических, так и «абстрактных» изображений-текстур. Алгоритмы, основанные на представленной модели,

обеспечивают генерацию и нанесение текстур в реальном времени и допускают аппаратную реализацию.

В заключении формулируются основные результаты работы.

Приложение содержит справочную информацию по основам теории вейвлет-анализа — многомасштабному анализу и вейвлет-преобразованиям непрерывных сигналов.

Благодарности

Автор выражает благодарность научному руководителю Ю.М. Баяковскому, а также Л.И. Левковичу-Маслюку (ИПМ им. М.В. Келдыша РАН) за сотрудничество и помощь в работе, А.В. Черницкому (ОАО «ВНИИнефть») за поддержку проекта по обработке линий уровня, компанию Intel Techologies, Inc. за поддержку проекта по генерации текстур.

Глава 1

Основы многомасштабного представления информации

1.1 Структура вейвлет-разложения сигнала

Как было отмечено во Введении, вейвлет-преобразование — это разложение сигнала по системе функций, которые являются сдвинутыми и масштабированными (сжатыми или растянутыми) копиями одной функции — *порождающего вейвлета*.

Правила выбора копий, участвующих в разложений, могут быть разными. Допустимо, например, разложение по всем возможным сдвигам и масштабам одной функции. Такое вейвлет-преобразование называется *непрерывным* [13, 45].

При наложении дополнительных условий на порождающий вейвлет, количество функций может быть сокращено до счетного множества. Наиболее распространенный случай — так называемое *диадное* преобразование, с помощью которого сигнал $f(x) \in \mathbf{L}_2(\mathbf{R})$ может быть представлен в виде

суммы следующего ряда:

$$f(x) = \sum_{i=-\infty}^{+\infty} \sum_{j=-\infty}^{+\infty} w_j^{(i)} \sqrt{2^i} \psi(2^i x - j). \tag{1.1}$$

Порождающий вейвлет — функция $\psi(x)$. Как правило, минимальными требованиями на порождающий вейвлет являются пространственная локализация (финитность, либо быстрое затухание на бесконечности) и наличие хотя бы одного нулевого момента, т.е. равенство нулю интеграла по всей области определения. Коэффициенты разложения $w_j^{(i)}$ называются *вейвлет-коэффициентами*. Индекс i является индексом масштаба и называется *уровнем разрешения* или *разрешением*; индекс j является индексом сдвига.

Для того, чтобы представление (1.1) существовало для любого $f(x) \in \mathbf{L}_2(\mathbf{R})$, необходимо, чтобы система функций, порожденная вейвлетом $\psi(x)$, являлась базисом в $\mathbf{L}_2(\mathbf{R})$ (более подробно о вейвлет-базисах см. в Приложении А).

Частичную сумму ряда (1.1)

$$f^{(i_0)}(x) \equiv \sum_{i=-\infty}^{i_0-1} \sum_{j=-\infty}^{+\infty} w_j^{(i)} \sqrt{2^i} \psi(2^i x - j), \quad i_0 \in \mathbf{Z},$$

называют приближением сигнала $f(x)$ с разрешением i_0. Сигнал $f(x)$ можно представить в виде суммы начального приближения (т.е. приближения с некоторым начальным разрешением i_0) и оставшихся членов ряда (1.1):

$$f(x) = f^{(i_0)}(x) + \sum_{i=i_0}^{+\infty} \sum_{j=-\infty}^{+\infty} w_j^{(i)} \sqrt{2^i} \psi(2^i x - j), \quad i_0 \in \mathbf{Z}. \tag{1.2}$$

Формула (1.2) хорошо иллюстрирует идеологию многомасштабного представления информации. Первый член суммы является грубым приближением сигнала. При добавлении к нему членов ряда, степень детализации

представления увеличивается, т.е. увеличивается разрешение, с которым представлен сигнал. Прослеживается аналогия представления (1.2) с тригонометрическим рядом Фурье. Однако в отличие от последнего, на каждом уровне разрешения в (1.2) имеется бесконечное число членов ряда, каждый из которых соответствует некоторому локальному участку области определения сигнала.

Возникает вопрос о вычислении вейвлет-коэффициентов. Как известно, коэффициенты разложения объекта по базису в некотором пространстве равны скалярным произведениям объекта на элементы базиса, который образует с исходным базисом биортогональную пару. Если базис ортонормированный, то он биортогонален самому себе. В этом смысле вейвлет-базис ничем не отличается от любого другого функционального базиса (в частности, вейвлет-базис может быть и ортонормированным), и вейвлет-коэффициенты можно считать с помощью скалярных произведений. Однако операция вычисления скалярного произведения достаточно дорогая. Кроме того, на практике приходится обрабатывать не непрерывные, а дискретные сигналы.

Существует достаточно эффективный способ вычисления коэффициентов разложения (предложенный С. Малла [55, 56]), использующий зависимости между вейвлет-коэффициентами разных уровней разрешения. Все вычисления производятся только над дискретными сигналами, при этом базисные функции (вейвлеты) в явном виде вообще не фигурируют.

Ниже этот способ рассматривается сначала на примере простейшего вейвлет-преобразования — преобразования Хаара, а затем в общем виде. Связь между таким методом расчета вейвлет-коэффициентов и разложени-

ем по вейвлет-базисам в $\mathbf{L}_2(\mathbf{R})$ обсуждается в Приложении А.

1.2 Преобразование Хаара

Пусть имеется одномерный дискретный сигнал $\mathbf{s} = \{s_j\}_{j\in\mathbf{Z}}$. Каждой паре элементов с индексами $2j$ и $2j+1$, $j \in \mathbf{Z}$, поставим в соответствие два значения:

$$\begin{aligned} v_j &= \frac{s_{2j} + s_{2j+1}}{2}, \\ w_j &= \frac{s_{2j} - s_{2j+1}}{2}. \end{aligned}$$

Эти значения формируют два новых сигнала $\mathbf{v} = \{v_j\}_{j\in\mathbf{Z}}$ и $\mathbf{w} = \{w_j\}_{j\in\mathbf{Z}}$, один из которых является огрубленной версией исходного сигнала (каждой паре элементов $\mathbf{s}$ соответствует их среднее арифметическое), а другой содержит информацию (будем называть ее *детализирующей*), необходимую для восстановления исходного сигнала. Действительно:

$$\begin{aligned} s_{2j} &= v_j + w_j, \\ s_{2j+1} &= v_j - w_j; \\ & j \in \mathbf{Z}. \end{aligned}$$

К сигналу $\mathbf{v}$ можно применить аналогичную операцию и так же получить два сигнала, один из которых является огрубленной версией $\mathbf{v}$, а другой содержит детализирующую информацию, необходимую для восстановления $\mathbf{v}$.

Поставим в соответствие сигналу $\mathbf{s}$ произвольный уровень разрешения i_1. Выпишем рекурсивные формулы для вычисления элементов сигналов

для любого разрешения $i_0 < i_1$:

$$v_j^{(i)} = \frac{v_{2j}^{(i+1)} + v_{2j+1}^{(i+1)}}{2},\ w_j^{(i)} = \frac{v_{2j}^{(i+1)} - v_{2j+1}^{(i+1)}}{2},$$
$$i = i_1 - 1,\ i_1 - 2,\ \ldots,\ i_0, \quad j \in \mathbf{Z}; \tag{1.3}$$
$$v_j^{(i_1)} = s_j, \quad j \in \mathbf{Z}.$$

Восстановление сигнала выполняется по формулам:

$$v_{2j}^{(i+1)} = v_j^{(i)} + w_j^{(i)},\ v_{2j+1}^{(i+1)} = v_j^{(i)} - w_j^{(i)}, \quad j \in \mathbf{Z},\ i = i_0,\ i_0 + 1,\ \ldots,\ i_1 - 1. \tag{1.4}$$

Формулы (1.3) и (1.4) определяют соответственно прямое и обратное *преобразование Хаара* одномерного дискретного сигнала. Преобразование Хаара является простейшим вейвлет-преобразованием.

1.3 Вейвлет-преобразования дискретных сигналов

На примере преобразования Хаара хорошо видна структура вейвлет-преобразования дискретного сигнала. На каждом шаге прямого преобразования сигнал распадается на две составляющие: приближение с более низким разрешением и детализирующую информация. Первую часто называют *низкочастотной* (НЧ), а вторую *высокочастотной* (ВЧ). Такая терминология принята по аналогии с анализом Фурье, однако не следует забывать, что частота в гармоническом анализе и частота в смысле уровня разрешения в вейвлет-анализе являются близкими, но не эквивалентными понятиями.

Элементы ВЧ составляющих вейвлет-преобразований называют *вейвлет-коэффициентами.*

Заметим, что шаг прямого преобразования Хаара эквивалентен свертке сигнала с фильтром (ядром) $\tilde{\mathbf{h}} = [1/2,\ 1/2]$ для НЧ составляющей и с филь-

тром $\tilde{\mathbf{g}} = [-1/2,\, 1/2]$ для ВЧ составляющей с последующим удалением из полученных сигналов каждого второго элемента. Это позволяет переписать формулы (1.3) следующим образом:

$$\begin{gathered}\mathbf{v}_i = \downarrow_2 \left[\mathbf{v}_{i+1} * \tilde{\mathbf{h}}\right],\ \mathbf{w}_i = \downarrow_2 \left[\mathbf{v}_{i+1} * \tilde{\mathbf{g}}\right], \quad i = i_1 - 1,\, i_1 - 2,\, \ldots,\, i_0; \\ \mathbf{v}_{i_1} = \mathbf{s}\end{gathered} \tag{1.5}$$

(удаление каждого второго элемента выполняет оператор $\downarrow_2 [\bullet]$).

Шаг обратного преобразования Хаара эквивалентен следующей последовательности операций. Между каждыми двумя элементами НЧ составляющей добавляется по одному нулевому элементу, аналогичным образом преобразуется и ВЧ составляющая. Далее производится свертка первого из полученных сигналов с фильтром $\mathbf{h} = [1,\, 1]$, а второго с фильтром $\mathbf{g} = [1,\, -1]$, результаты поэлементно складываются. Формула (1.4) переписывается следующим образом:

$$\mathbf{v}_{i+1} = \uparrow_2 [\mathbf{v}_i] * \mathbf{h} + \uparrow_2 [\mathbf{w}_i] * \mathbf{g}, \quad i = i_0,\, i_0 + 1,\, \ldots,\, i_1 - 1, \tag{1.6}$$

(вставку нулевых элементов выполняет оператор $\uparrow_2 [\bullet]$).

В формулы (1.5) и (1.6) являются формулами соответственно прямого и обратного вейвлет-преобразований дискретных сигналов в общем виде. Конкретный вид преобразования задается четверкой фильтров $\tilde{\mathbf{h}}$, $\tilde{\mathbf{g}}$, $\mathbf{h}$ и $\mathbf{g}$ (как это было сделано выше для преобразования Хаара), которые называются соответственно НЧ фильтром анализа, ВЧ анализа, НЧ синтеза и ВЧ синтеза.

Естественно, фильтры должны быть подобраны так, чтобы преобразование было обратимым, то есть

$$\uparrow_2 \left[\downarrow_2 \left[\mathbf{v} * \tilde{\mathbf{h}}\right]\right] * \mathbf{h} + \uparrow_2 [\downarrow_2 [\mathbf{v} * \tilde{\mathbf{g}}]] * \mathbf{g} \equiv \mathbf{v}.$$

Часто формулы (1.5) и (1.6) записывают с помощью так называемого Z-преобразования[1]. Свертка сигналов эквивалентна произведению их Z-преобразований (свойство, которым обладает и преобразование Фурье).

Для Z-преобразования некоторого сигнала $\mathbf{s}$ будем использовать обозначение $s(z)$.

Вот как выглядят формулы (1.5) и (1.6) в терминах z-преобразований:

$$\begin{gathered} v_i(z) = \downarrow_2 \left[\tilde{h}(z)\, v_{i+1}(z)\right],\ w_i(z) = \downarrow_2 \left[\tilde{g}(z)\, v_{i+1}(z)\right], \\ i = i_1 - 1,\, i_1 - 2,\, \ldots,\, i_0; \\ v_i(z) = s(z). \end{gathered} \tag{1.7}$$

$$v_{i+1}(z) = h(z) \cdot \uparrow_2 [v_i(z)] + g(z) \cdot \uparrow_2 [w_i(z)], \quad i = i_0,\, i_0 + 1,\, \ldots,\, i_1 - 1. \tag{1.8}$$

В терминах Z-преобразований легко формулируются условия, которым должны удовлетворять фильтры, чтобы преобразование было обратимым:

$$\begin{aligned} h(z)\,\tilde{h}(z) + g(z)\,\tilde{g}(z) &= 2, \\ h(z)\,\tilde{h}(-z) + g(z)\,\tilde{g}(-z) &= 0. \end{aligned} \tag{1.9}$$

Если, кроме того, $\tilde{h}(z) = C\,h(z^{-1})$ и $\tilde{g}(z) = C\,g(z^{-1})$, где C — некоторая константа, то преобразование называется *ортогональным* (это преобразование соответствует разложению сигнала по ортогональному или, если $C \equiv 1$, ортонормированному вейвлет-базису), в противном случае оно называется *биортогональным*.

Примеры. Преобразование Хаара является ортогональным. Фильтры

[1] *Z-преобразованием* дискретного сигнала $\mathbf{s} = \{s_j\}_{j \in Z}$ называется полином Лорана $P_{\mathbf{s}}(z) = \sum_{j \in \mathbf{Z}} s_j\, z^{-j}$.

Хаара в форме Z-преобразований имеют вид:

$$\tilde{h}(z) = \frac{1}{2}(z^{-1} + 1), \;\; \tilde{g}(z) = \frac{1}{2}(-z^{-1} + 1),$$

$$h(z) = (1 + z), \qquad g(z) = (1 - z).$$

Ортогональное вейвлет-преобразование Добеши D_4 (число 4 обозначает количество ненулевых коэффициентов в фильтрах):

$$\tilde{h}(z) = h(z^{-1}), \quad h(z) = h_0 + h_1 z^{-1} + h_2 z^{-2} + h_3 z^{-3},$$

$$\tilde{g}(z) = g(z^{-1}), \quad g(z) = -h_3 z^2 + h_2 z - h_1 + h_0 z^{-1};$$

$$h_0 = \tfrac{1+\sqrt{3}}{4\sqrt{2}}, \quad h_1 = \tfrac{3+\sqrt{3}}{4\sqrt{2}}, \quad h_2 = \tfrac{3-\sqrt{3}}{4\sqrt{2}}, \quad h_3 = \tfrac{1-\sqrt{3}}{4\sqrt{2}}.$$

Пример биортогонального преобразования (кубическое B-сплайновое вейвлет-преобразование) можно найти в гл. 3.

1.4 Вейвлет-преобразования конечных сигналов

При обработке конечных сигналов могут возникнуть проблемы с вычислением преобразований вблизи границ сигнала.

Простейший случай — преобразование Хаара сигнала, количество элементов которого равно степени двух. Пусть $\mathbf{s} = \{s_j\}_{j=0}^{K-1}$ для некоторого $K = 2^{i_1}$, $i_1 \in \mathbf{Z}$, $i_1 > 0$. Число i_1 будем считать разрешением сигнала $\mathbf{s}$. На каждом шаге прямого преобразования НЧ и ВЧ составляющие будут получаться в 2 раза короче преобразуемого сигнала. После выполнения i_1 шагов прямого преобразования НЧ и ВЧ составляющие будут иметь по одному элементу (им соответствует разрешение $i_0 = 0$). При вычислении коэффициентов преобразования не происходит выхода за границы сигнала,

поэтому для вычисления преобразования не требуется никаких дополнительных действий с фильтрами или сигналом.

Если же длина фильтра больше 2, то при вычислении свертки должен произойти выход за границы сигнала. Это приводит к необходимости либо менять фильтр вблизи границ сигнала (что усложняет алгоритм преобразования), либо доопределять сигнал за его границами (нулями, симметрично, периодически и пр.), но это может в общем случае потребовать сохранения дополнительной информации, необходимой для полного восстановления сигнала. Сохранение дополнительной информации может потребоваться и при обработке сигнала нечетной длины.

Проблема обработки сигналов вблизи границ рассматривается в задаче построения изолиний (гл. 3). Более подробно о преобразовании конечных сигналов см. в [13].

1.5 Вейвлет-преобразования двумерных сигналов

Рассмотрим двумерный сигнал $\mathbf{s}$ — матрицу конечного или бесконечного размера. Применим к каждой строке матрицы один шаг одномерного вейвлет-преобразования. В результате получится две матрицы, строки которых содержат НЧ и ВЧ составляющие строк исходной матрицы. К каждому столбцу обеих матриц также применим шаг одномерного преобразования. В результате получается четыре матрицы, которые можно обозначить как НЧ/НЧ, НЧ/ВЧ, ВЧ/НЧ и ВЧ/ВЧ. Первая является НЧ составляющей исходного сигнала, остальные три содержат детализирующую информацию.

Таким образом, шаг двумерного преобразования свелся к композиции одномерных преобразований. Поэтому реализация двумерного преобразова-

ния не требует никаких дополнительных операций. В частности, в случае конечных сигналов действуют те же правила, что и при преобразовании одномерных конечных сигналов.

Композиция одномерных преобразований — это не единственная схема вейвлет-преобразования сигналов больших размерностей (хотя и наиболее распространенная). Подробнее о многомерных преобразованиях см. [13, 30].

1.6 Древовидные структуры для представления вейвлет-преобразований

Рассмотрим одномерное диадное вейвлет-преобразование дискретных сигналов. Как было отмечено выше, один шаг прямого преобразования ставит в соответствие каждой паре элементов сигнала один элемент НЧ составляющей и один вейвлет-коэффициент (рис. 1.1, слева). Пусть исходному сигналу приписан уровень разрешения i_1. Для каждой пары соседних элементов этого сигнала построим двоичное дерево глубины 1, корень которого соответствует элементу НЧ составляющей уровня $i_1 - 1$, а две листовых вершины — паре исходных элементов. К корневой вершине припишем вейвлет-коэффициент уровня $i_1 - 1$.

Следующий шаг ставит в соответствие каждой паре элементов НЧ составляющей уровня $i_1 - 1$ один НЧ и один ВЧ элемент уровня $i_1 - 2$. Соответствующие пары деревьев объединяются новой корневой вершиной (следовательно, глубина деревьев вырастает до 2), которая соответствует НЧ элементу уровня $i_1 - 2$, и ей приписывается соответствующий вейвлет-коэффициент уровня $i_1 - 2$. Аналогичным образом сращиваются пары деревьев для получение уровня $i_1 - 3$ и так далее. Если исходный сигнал бесконечный,

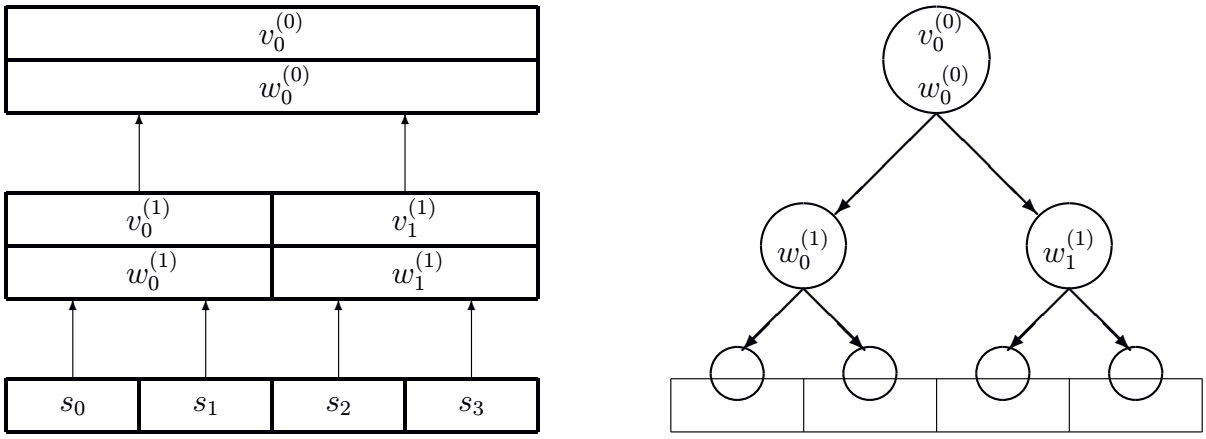


Рис. 1.1: Структура вейвлет-преобразования (слева) и дерево преобразования (справа).

то процесс можно прервать на любом уровне i_0, $i_0 < i_1$, в результате чего получится лес бесконечного числа сбалансированных двоичных деревьев глубины $i_1 - i_0$. Если сигнал конечен (для простоты рассмотрим случай, когда длина исходного сигнала равна 2^{i_1}), то по выполнении i_1 шагов прямого преобразования НЧ составляющая уровня 0 будет иметь единственный элемент, а вместо леса получится единственное дерево глубины i_1.

По окончании процесса преобразования всем корневым вершинам (или одной вершине, если построено единственное дерево) приписываются, кроме вейвлет-коэффициентов, еще и соответствующие НЧ элементы. Таким образом, корневым вершинам приписывается по 2 значения, а листовым — ни одного (рис. 1.1, справа).

При выполнении обратного преобразования движение по дереву происходит от корня к листовым вершинам.

В двумерном случае на каждом шаге прямого преобразования один элемент НЧ составляющей соответствует четырем элементам исходного сигнала (образующим матрицу размера 2×2), при этом образуется три ВЧ

составляющих. Следовательно, дерево двумерного преобразования является квадродеревом (4 потомка у каждой нелистовой вершины), и каждой нелистовой вершине приписывается по 3 вейвлет-коэффициента.

Древовидное представление хорошо демонстрирует пространственную локализацию преобразования. Пусть исходному сигналу соответствует некоторая пространственная область. Любая вершина дерева соответствует некоторому фрагменту этой области. Чем дальше от корня выбрана вершина, тем меньшему фрагменту она соответствует. Все поддерево, растущее из этой вершины, соответствует тому же фрагменту, за исключением, быть может, участков вблизи границ фрагмента.

Древовидная структура, кроме наглядной иллюстрации структуры преобразуемого сигнала, нашла практическое применение в задаче сжатия информации. Основная идея сжатия с помощью вейвлет-преобразования состоит в том, чтобы исключить из вейвлет-образа коэффициенты, абсолютные величины которых близки к нулю. Возникает вопрос, как кодировать такое представление. Один из способов был предложен в работе [68]. Было сделано предположение, что если вейвлет-коэффициент (или, в многомерном случае, коэффициенты), приписанный некоторой нелистовой вершине дерева преобразования, имеет пренебрежимо малую абсолютную величину, то, не внося существенной ошибки в восстанавливаемый сигнал, можно обратить в нуль и все вейвлет-коэффициенты поддерева, растущего из этой вершины. Такое поддерево получило название *нуль-дерева*. Очевидно, что для кодирования нуль-дерева достаточно лишь запомнить одну вершину, которая является его корнем. Нуль-деревья используются во многих алгоритмах вейвлет-сжатия, например, [66].

Глава 2

Адаптивное сеточное представление объектов, определенных на плоскости. Задача реконструкции освещенности на плоскости

2.1 Двумерные сигналы и сеточное представление

В компьютерной графике широкий класс объектов представим с помощью двумерных дискретных сигналов (т.е. функций, определенных на плоскости и оцифрованных с некоторым конечным разрешением).

Наиболее очевидный пример — растровые изображения (битовые матрицы). Не менее распространенный объект — поверхность, которая является графиком функции двух переменных. С помощью такой поверхности изображается, например, ландшафт в играх и симуляторах. Хранить поверхность в виде двумерного массива — карты высот — нерентабельно. Как правило, такая поверхность задается сеткой, треугольной или четы-

рехугольной.

Сетка, построенная непосредственно по исходному двумерному массиву, не дает практически никаких преимуществ перед исходным представлением. Интересна задача построения так называемой *адаптивной* сетки, т.е. сетки, построенной на основе анализа структуры конкретного объекта. Такая сетка должна содержать возможно меньше вершин, при этом исходный объект по такой сетке должен быть восстановим либо без потерь, либо с контролируемой пользователем погрешностью.

Существуют алгоритмы адаптивной триангуляции или прореживания, основанные на том, что из частой треугольной сетки удаляются «малосущественные» вершины (критерии существенности могут быть различными), и в тех местах, откуда вершины были удалены, производится повторная локальная триангуляция [67]. Такие алгоритмы как правило довольно сложны, а их реализации дороги в смысле объема вычислений.

2.2 Использование вейвлет-анализа для построения адаптивных сеток

Частотно-пространственная структура многомасштабного представления объекта позволяет предположить, что на основе такого представления возможно эффективное построение адаптивной сетки. При этом используются почти такие же соображения, что и при сжатии информации.

Существует два принципиальных подхода к созданию адаптивных сеток на основе многомасштабного представления объекта. Первый — это построение многомасштабного анализа непосредственно на самой сетке [53, 54, 73]. Такая сетка должна иметь специальную структуру, которая допус-

кает многомасштабный анализ. Шаги прямого и обратного преобразования делают эту сетку более крупной или, наоборот, частой. При этом по величине вейвлет-коэффициентов можно определить, насколько измельченной должна быть сетка для адекватного представления того или иного фрагмента объекта. Достоинством такого метода является его универсальность. Исходную сетку можно построить для любой поверхности в пространстве (а не только для поверхностей, параметризуемых на плоскости, о которых идет речь в данной главе). Однако подход имеет и недостатки. Прежде всего, построение исходной сетки, допускающей многомасштабный анализ, является в общем случае нетривиальной и трудоемкой задачей [32]. Кроме того, на сетках применяются не обычные многомерные преобразования, полученные композицией одномерных, а преобразования на специальных решетках (о преобразованиях на решетках в общем виде см. в [13, 49]), которые требуют более сложной реализации.

Второй подход к построению адаптивных сеток — сначала найти те вершины, по которым будет построена итоговая сетка, а потом построить сетку по найденным вершинам.

Для поиска вершин предлагается воспользоваться древовидным представлением, описанным в гл. 1, п. 1.6. Если построить дерево преобразования и удалить из него все нуль-деревья, возникшие в результате отбрасывания части вейвлет-коэффициентов, то полученное дерево (вообще говоря, несбалансированное) будет отражать структуру особенностей рассматриваемой поверхности.

Рассмотрим вопрос о том, как на основе дерева построить сетку.

2.2.1 Дерево узлов

Пусть в качестве исходных данных имеется дискретный конечный сигнал $\mathbf{s} = \{s_{j,k}\}$, $j = \overline{0,\, J-1}$, $k = \overline{0,\, K-1}$. Этот сигнал есть результат оцифровки поверхности, выполненной с некоторым шагом. Следовательно, элементы сигнала суть значения в узлах регулярной решетки. Будем считать, что поверхность рассматривается на прямоугольной области $[0,\, J-1] \times [0,\, K-1]$, тогда координатами узлов решетки на плоскости можно считать их индексы, то есть узел с индексами (j, k) имеет координаты $x_{j,k} = j$, $y_{j,k} = k$ и значение $s_{j,k}$.

Для простоты изложения будем считать, что $J = K = 2^{i_1}$. В этом случае сбалансированное квадродерево глубины i_1 будет иметь ровно столько листовых вершин, сколько элементов имеет сигнал $\mathbf{s}$.

К сигналу применяется обычное двумерное вейвлет-преобразование, описанное в гл. 1, п. 1.5.

Положим $\mathbf{v}_{i_1} \equiv \mathbf{s}$. Каждому элементу этого сигнала соответствует одна листовая вершина дерева преобразований. Решетку, соответствующую $\mathbf{v}_{i_1}$, будем называть, по аналогии с сигналом, решеткой уровня разрешения i_1. Для координат узлов решетки также воспользуемся обозначениями, аналогичными обозначениям элементов сигнала:

$$\mathbf{x}_{i_1} = \{x^{(i_1)}_{j,k} \equiv x_{j,k}\},\ \mathbf{y}_{i_1} = \{y^{(i_1)}_{j,k} \equiv y_{j,k}\}; \quad j,k \in \mathbf{Z},\ 0 \le j,k < 2^{i_1}.$$

Выполнив один шаг преобразования, получим сигнал $\mathbf{v}_{i_1-1}$. В дереве преобразований элементам этого сигнала соответствуют вершины уровня $i_1 - 1$. Связи между вершинами уровней $i_1 - 1$ и i_1 показывают, каким четырем элементам сигнала $\mathbf{v}_{i_1}$ соответствует каждый элемент сигнала $\mathbf{v}_{i_1-1}$.

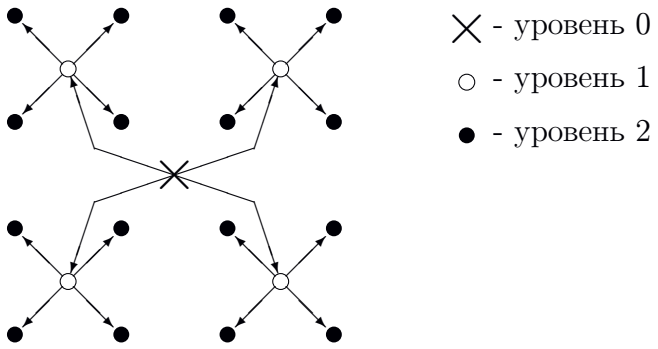


Рис. 2.1: Узлы решеток уровней 0, 1 и 2 (стрелками показаны ребра дерева).

Сигнал $\mathbf{v}_{i_1-1}$ также является результатом оцифровки той же поверхности, но выполненной с увеличенным вдвое шагом решетки. Координаты узлов этой решетки (разрешения i_1-1) считаются как среднее арифметическое координат четырех соответствующих ему узлов решетки разрешения i_1, то есть

$$x_{j,k}^{(i_1-1)} = \frac{x_{2j,2k}^{(i_1)} + x_{2j+1,2k}^{(i_1)}}{2},\ y_{j,k}^{(i_1-1)} = \frac{y_{2j,2k}^{(i_1)} + y_{2j,2k+1}^{(i_1)}}{2};\ j,k \in \mathbf{Z},\ 0 \le j,k < 2^{i_1-1}.$$

Аналогичным образом строятся решетки разрешений от i_1-2 до 0, (на рис. 2.1 показано взаимное расположение узлов решеток разрешений 0, 1 и 2). Значения в узлах вычисляются по формулам прямого вейвлет-преобразования.

Получается, что любая вершина дерева преобразований соответствует одному узлу какой-либо решетки. Каждый узел, как и каждая вершина дерева, имеет свой уникальный тройной индекс (разрешение i и двойной индекс на плоскости (j,k)).

Теперь сбалансированное дерево преобразований требуется превратить в несбалансированное за счет удаления нуль-деревьев. Критерий определения нуль-деревьев может быть разным. Простейший случай — установить

порог абсолютной величины вейвлет-коэффициента. Если абсолютная величина коэффициента, приписанного некоторой вершине дерева, оказалось ниже уровня порога, то этот коэффициент обращается в нуль, то же самое происходит и со всеми вейвлет-коэффициентами, которые приписаны к поддереву, растущему из данной вершины. Заметим, что в двумерном случае каждой нелистовой вершине приписано не по одному, а по три вейвлет-коэффициента. Можно сравнить сумму квадратов тройки коэффициентов с квадратом величины порога и либо обращать в нуль все три коэффициента, если сумма окажется меньше, либо, в противном случае, оставлять все коэффициенты без изменения.

Листовые вершины полученного несбалансированного дерева определяют те узлы, по которым будет строиться окончательная сетка. Благодаря свойству пространственной локализации вейвлет-преобразования, расположение узлов адаптировано к особенностям конкретной поверхности. Их плотность высока там, где поверхность имеет особенности, и низка в тех местах, где поверхность изменяется незначительно. Остается рассмотреть вопрос, как по полученному набору узлов построить треугольную сетку.

Существует метод построения треугольной сетки по набору произвольно расположенных узлов, известный как триангуляция Делоне. Однако в данном случае этот метод решено не применять ввиду его достаточно сложной реализации.

Простой метод триангуляции можно построить на основе все того же дерева преобразований.

Создается двумерный массив ссылок на вершины, который имеет в точности такой же размер, что и решетка уровня i_1. Каждый элемент массива

соответствует одному узлу этой решетки. Элементы массива заполняются следующим образом: если узлу решетки соответствует листовая вершина дерева, то в соответствующий узлу элемент массива заносится ссылка на эту вершину; если узлу решетки соответствует вершина нуль-дерева, то в элемент массива заносится ссылка на корень этого нуль-дерева. Таким образом массив заполняется ссылками на листовые вершины дерева, причем часть элементов массива содержит ссылки на одни и те же вершины (кроме случая, когда нуль-деревья отсутствуют).

Далее по полученному массиву строится триангуляция, как если бы триангулировалась регулярная решетка уровня i_1. Для каждой пары индексов (j, k), $0 \le j, k < 2^{i_1}$, строится по два треугольника, на вершины которых указывают элементы массива с индексами (j, k), $(j+1,\ k)$, $(j,\ k+1)$ и $(j+1,\ k+1)$. Это можно сделать двумя способами, соединив по диагонали две разных пары противолежащих вершин, как показано на рис. 2.2.

При этом окажется, что часть треугольников будет иметь по две или по три совпадающие вершины (за счет повторяющихся ссылок в массиве). Такие треугольники просто игнорируются. Оставшиеся же треугольники и образуют искомую сетку (рис. 2.3).

Подробнее рассмотрим вопрос о том, какую из представленных на рис. 2.2 возможностей построения двух треугольников по четырем вершинам предпочесть. Проще всего зафиксировать один вариант и строить согласно ему всю триангуляцию. Можно чередовать варианты, либо выбирать случайным образом с вероятностью 1/2. Все эти способы не учитывают особенностей конкретной поверхности. Однако есть и подходы, учитывающие расположение вершин треугольников и значения в них. Например,

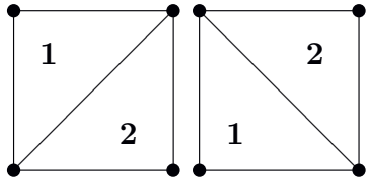
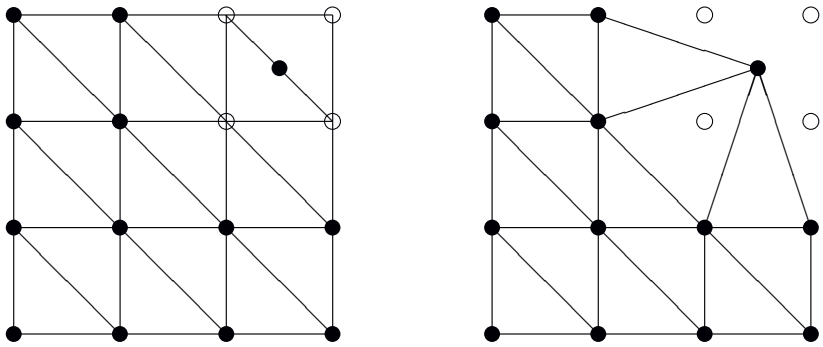


Рис. 2.2: Два способа построения треугольников по четырем вершинам.



● - узлы, соотв. листовым вершинам дерева

○ - узлы, соотв. некорневым вершинам нуль-деревьев

Рис. 2.3: Триангуляция массива ссылок (слева) и итоговая сетка (справа).

выбирать вариант, при котором длина смежной стороны пары треугольников будет наименьшей из двух возможных (причем длину стороны можно вычислять как на плоскости, так и в пространстве, учитывая значения в вершинах). Этот подход позволит сократить число тупоугольных треугольников в сетке, заменяя их остроугольными. Другой подход — выбирать вариант, при котором модуль разности значений несмежных вершин наибольший из двух возможных. Если поверхность достаточно резко меняет значение в области, ограниченной соответствующей парой треугольников, то в этом случае смежная сторона треугольников пройдет вдоль перепада, а не поперек.

Возникает вопрос о том, с помощью какого базиса выполнять преобразо-

вание. Очевидно, что дереву преобразований наилучшим образом соответствует преобразование Хаара. Преобразование Хаара может вычисляться независимо для любого поддерева. Как следствие, обратное преобразование можно выполнять не для всего дерева, а только для поддеревьев, которые не являются нуль-деревьями. Кроме того, двумерное преобразование Хаара можно реализовывать не по общей схеме композиций двух одномерных преобразований, а по следующим формулам:

$$\begin{aligned}
v_{j,k}^{(i)} &= \frac{v_{2j,2k}^{(i+1)} + v_{2j+1,2k}^{(i+1)} + v_{2j,2k+1}^{(i+1)} + v_{2j+1,2k+1}^{(i+1)}}{4}, \\
vw_{j,k}^{(i)} &= \frac{v_{2j,2k}^{(i+1)} + v_{2j+1,2k}^{(i+1)} - v_{2j,2k+1}^{(i+1)} - v_{2j+1,2k+1}^{(i+1)}}{4}, \\
wv_{j,k}^{(i)} &= \frac{v_{2j,2k}^{(i+1)} - v_{2j+1,2k}^{(i+1)} + v_{2j,2k+1}^{(i+1)} - v_{2j+1,2k+1}^{(i+1)}}{4}, \\
ww_{j,k}^{(i)} &= \frac{v_{2j,2k}^{(i+1)} - v_{2j+1,2k}^{(i+1)} - v_{2j,2k+1}^{(i+1)} + v_{2j+1,2k+1}^{(i+1)}}{4};
\end{aligned}$$

$$i = i_1 - 1,\ i_1 - 2,\ \ldots,\ 0; \quad j, k \in \mathbf{Z},\ 0 \le j, k < 2^{i_1}.$$

Обратное преобразование выполняется по формулам:

$$\begin{aligned}
v_{2j,2k}^{(i+1)} &= v_{j,k}^{(i)} + vw_{j,k}^{(i)} + wv_{j,k}^{(i)} + ww_{j,k}^{(i)}, \\
v_{2j+1,2k}^{(i+1)} &= v_{j,k}^{(i)} + vw_{j,k}^{(i)} - wv_{j,k}^{(i)} - ww_{j,k}^{(i)}, \\
v_{2j,2k+1}^{(i+1)} &= v_{j,k}^{(i)} - vw_{j,k}^{(i)} + wv_{j,k}^{(i)} - ww_{j,k}^{(i)}, \\
v_{2j+1,2k+1}^{(i+1)} &= v_{j,k}^{(i)} - vw_{j,k}^{(i)} - wv_{j,k}^{(i)} + ww_{j,k}^{(i)};
\end{aligned}$$

$$i = 0,\ 1,\ \ldots,\ i_1 - 1; \quad j, k \in \mathbf{Z},\ 0 \le j, k < 2^{i_1}.$$

Очевидно, что, хотя вейвлеты Хаара являются кусочно-постоянным функциями, поверхность, состоящая из треугольников, вершины которых вычислены с помощью этого преобразования, является кусочно-билинейной.

Простое в реализации преобразование Хаара обладает рядом известных недостатков. В частности, вейвлеты Хаара имеют весьма ограниченные

шумопонижающие возможности. Поэтому наряду с ним приходится использовать и другие преобразования, базисные функции которых более гладкие и имеют по крайней мере 2 нулевых момента, например, Добеши D_4 или линейные сплайны.

Использование базисов, функции которых имеют бóльшую гладкость и большее число нулевых моментов, предполагает, что на треугольниках итоговой сетки будут строиться фрагменты поверхностей более высоких порядков, например, кубические B-сплайны. Это может сократить количество треугольников, но усложнит процедуру генерации результата.

Следует также отметить, что преобразование по любому вейвлет-базису, кроме базиса Хаара, приходится выполнять по общей схеме и для каждого из уровней разрешения целиком, а не для отдельных поддеревьев (при этом все вейвлет-коэффициенты, соответствующие нуль-деревьям, считаются равными нулю). Это необходимо потому, что при вычислении преобразований, задаваемых фильтрами длиной более 2, происходит выход за границы областей, соответствующих ненулевым поддеревьям дерева преобразований.

2.2.2 Альтернативный подход: дерево ячеек

Предложенный подход, очевидно, не является единственно возможным. Например, в работе [39] описан другой вариант построения треугольных сеток на основе вейвлет-анализа и квадродеревьев. Вершинам дерева соответствуют не узлы решетки, а квадратные ячейки, вершинами которых являются узлы (назовем такое дерево деревом ячеек). На максимальном уровне разрешения (дерево сбалансировано) все ячейки одинакового размера и не

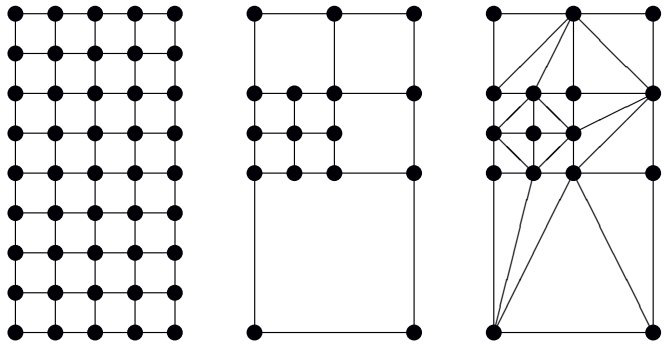


Рис. 2.4: Построение сетки с помощью дерева ячеек.

имеют вершин на ребрах, кроме угловых (рис. 2.4, слева).

Несбалансированному дереву соответствуют ячейки разных размеров — такое становится возможным благодаря удалению части узлов, определение таких узлов производится с помощью вейвлет-анализа. Размеры сторон ячеек отличаются друг от друга в число раз, равное степени 2. При этом часть ячеек получает дополнительные вершины на ребрах, если эти ячейки граничат с ячейками меньшего размера (рис. 2.4, в центре).

Триангуляция производится отдельно в каждой ячейке. Если дополнительно ограничить максимальное количество неугловых вершин на ребрах, то число возможных комбинаций узлов в ячейке будет конечным. (Авторы предлагают ограничить количество неугловых вершин тремя. Это значит, что размеры сторон соседних ячеек могут отличаться не более чем в 4 раза.) Поскольку количество комбинаций конечно, то составляется словарь комбинаций, и триангуляция выполняется с его помощью (рис. 2.4, справа).

Подход, основанный на дереве ячеек, близок к алгоритмам адаптивной триангуляции в том смысле, что формирует новую сетку, прореживая исходную, не меняя местоположений вершин и их значений. Это может быть полезно, например, в тех случаях, когда требуется, чтобы определенные

точки строящегося объекта имели одни и те же значения вне зависимости от уровня детализации, значения допустимой погрешности аппроксимации и т.д. С другой стороны, нетрудно доработать метод так, чтобы, в случае необходимости, значения в узлах могли меняться. Метод, использующий дерево вершин, строит новую сетку, интерполирующую исходные данные, но ее вершины не являются, вообще говоря, подмножеством вершин исходной сетки.

Метод, основанный на дереве ячеек, имеет, на наш взгляд, один существенный недостаток. Ни одна из возможных схем разбиения решетки на подрешетки (см. [49]) не укладывается полностью в предложенное разбиение на ячейки. Авторы пользуются неким урезанным вариантом «шахматной доски», который не исчерпывает всех узлов исходной решетки. Другими словами, часть узлов исходной решетки не рассматривается ни на одном из уровней разрешения. Судя по примерам, представленным авторами работы, эти узлы отбрасываются, как только отбрасываются все их соседи, попавшие в один из уровней разрешения. Такое «неравенство» не вполне корректно с математической точки зрения.

Предложенный выше метод, основанный на дереве узлов, также не лишен недостатков. Прежде всего, из-за того, что вершины меняют свои местоположения, затрудняется обработка сигнала вблизи границ, так как граничные вершины также могут сдвигаться внутрь рассматриваемой области, в связи с чем происходит уменьшение последней. Чтобы бороться с этим, приходится либо запрещать отсечение нуль-деревьев вблизи границ, что существенно увеличивает количество треугольников итоговой сетки, либо специальным образом сдвигать вершины в пограничных треугольниках

(при этом может потребоваться корректировка значений в этих вершинах). Метод, основанный на дереве ячеек, лишен этого недостатка.

С точки зрения реализации, построение дерева узлов более простое, чем построение дерева ячеек, однако в последнем случае более эффективной оказывается фаза триангуляции.

2.2.3 Примеры работы метода

На с. 111 показаны некоторые результаты применения описанного метода построения адаптивных сеток. Размер исходной регулярной решетки 64×64, сетка состоит из 7938 треугольников (рис. I.1, слева). К решетке применяются 2 шага ненормализованного преобразования Хаара. Критерий удаления нуль-деревьев — порог величины вейвлет-коэффициента. Значения в узлах находятся в диапазоне [0, 1], в этом же диапазоне определяется и порог.

На рис. I.1, справа, показана решетка, соответствующая нулевому порогу. Это значит, что при построении сетки не произошло потери информации. Сетка содержит 2046 узлов и 4120 треугольников (около половины от исходного количества).

На рис. I.2 показаны сетки, полученные в результате увеличения значения порога. Сетка справа: порог 0.1, узлов 1651, треугольников 3252. Сетка слева: порог 0.2, узлов 1396, треугольников 2744.

2.3 Многомасштабный анализ и реконструкция освещенности

Рассмотрим вопрос о том, как метод, основанный на дереве узлов, может быть применен при построение адаптивных сеток в задачах синтеза реалистических изображений.

2.3.1 Методы глобальной освещенности синтеза фотореалистических изображений

Синтез сложных трехмерных сцен с фотореалистическим качеством является одной из важных задач компьютерной графики.

Для синтеза подобных объектов используются так называемые методы глобальной освещенности. Эти методы объединяет идея моделирования физических процессов распространения света в пространстве.

Существуют три основных метода глобальной освещенности: излучательность, прямая трассировка лучей и обратная трассировка лучей.

Первый метод основан на моделировании процесса распространения световой энергии в замкнутой системе (которой является сцена) и сводится к решению уравнения энергетического баланса. Методы трассировки лучей моделируют распространение лучей света на основании законов геометрической оптики. Прямая трассировка моделирует распространение лучей от источников света, обратная трассировка моделирует «обратный ход» лучей, т.е. от наблюдателя. Каждый из методов имеет свои достоинства и недостатки. Так, излучательность применима только для диффузных[1] по-

[1] *Диффузной* или *ламбертовой* называется поверхность, равномерно отражающая падающий свет во всех направлениях (например, матовое стекло).

верхностей; прямая трассировка применима для поверхностей с более сложными отражающими свойствами, но неэффективна для моделирования зеркального отражения; обратная трассировка идеальна для зеркальных[2] поверхностей, но не применима для поверхностей диффузных. Излучательность и прямая трассировка выполняются вне зависимости от наблюдателя. Обратная трассировка, наоборот, жестко привязана к наблюдателю, и при изменении его местоположения требует полного пересчета всей сцены. Зато для каждого конкретного положения обратная трассировка практически не требует «лишних» вычислений, поскольку оперирует только с лучами, идущими к наблюдателю, т.е. только с видимыми им лучами, в то время как при прямой трассировки бóльшая часть вычислений оказывается лишней для синтеза сцены с точки зрения конкретного наблюдателя.

Таким образом, на практике в настоящее время применяются различные комбинированные подходы. Как правило расчет сцены происходит в два этапа: сначала считается не зависящая от наблюдателя фаза для незеркальных поверхностей (с помощью излучательности или разновидности прямой трассировки), затем для каждого конкретного положения наблюдателя на основе обратной трассировки производятся расчеты для зеркальных поверхностей.

Решение уравнения излучательности является достаточно сложной математической задачей, численные алгоритмы весьма нетривиальны, объемы вычислений велики. Алгоритм, реализующий трассировку лучей гораздо более прост, наиболее сложная его часть — это поиск пересечений

[2]Поверхность, идеальная с точки зрения геометрической оптики, т.е. поверхность, для которой падающий и отраженный лучи лежат в плоскости, перпендикулярной плоскости поверхности, и угол падения равен углу отражения.

прямой (луча) с объектами различной формы, однако объем вычислений также велик (по причине большого количества лучей), в особенности для случая прямой трассировки.

2.3.2 Метод Монте-Карло трассировки лучей

Один из путей сокращения объема вычислений при расчете освещенности — применение метода стохастических испытаний, известного также как метод Монте-Карло [47, 69, 70]. Ниже этот метод рассматривается на примере прямой трассировки лучей.

Светопередающие свойства объектов сцены можно описать в виде вероятностных распределений. Для источников света задается вероятность угла испускания светового луча. Например, для идеального точечного источника испускание луча во всех направлениях равновероятно, для направленного источника луч с наибольшей вероятностью должен попасть в конус с заданными характеристиками. Для неизлучающих поверхностей задается вероятность направления отраженного и/или преломленного луча в зависимости от направления луча падающего. Например, для идеальной зеркальной поверхности угол отражения должен быть с вероятностью 1 равен углу падения, а для диффузной поверхности вне зависимости от угла падения луч с равной вероятностью отражается в любом направлении. Также задается вероятность поглощения луча.

Процесс трассировки заключается в испускании конечного числа лучей из имеющихся в сцене источников света и моделировании их распространения с учетом описанных свойств источника и поверхностей. Все лучи имеют одинаковую неделимую мощность. Количество лучей, выпущенных

из каждого источника света, должно быть пропорционально мощности этого источника. Можно считать, что один такой луч эквивалентен одному кванту света (или фотону). Для каждой поверхности создается специальная структура данных, в которой регистрируется каждое попадание луча на поверхность [46, 47, 70]. В простейшем случае это массив точек (фотонов) в системе координат, привязанной к данной поверхности.

Освещенность поверхности — это функция плотности фотонов на поверхности. Очевидно, что чем больше фотонов попадет на поверхность, тем точнее можно восстановить функцию плотности.

Несомненным достоинством метода является его простота. В основе лежит простая и понятная идея, не требующая сложных математических выкладок, реализация алгоритма также не должна доставлять особых проблем.

Однако нужно отметить существенный недостаток, характерный для всех методов, в основе которых лежит идея приближения непрерывного процесса конечным числом стохастических экспериментов, — это медленная сходимость. Для удовлетворительного восстановления функции освещенности по фотонной карте количество фотонов на ней должно быть достаточно велико, но и это не может полностью гарантировать качественное восстановление функции освещенности — результат может оказаться искажен шумом. Для восстановления освещенности по возможно меньшему количеству фотонов требуется применение специальных методов.

2.3.3 Представление функции освещенности

Рассмотрим вопрос о том, в каком виде обычно требуется получить результат — функцию освещенности. Два основных способа представления этой функции — текстуры и сетки. Текстура — это растровое изображение, которое наносится на поверхность соответствующего объекта. С точки зрения структуры данных текстура — это матрица (двумерный массив) значений функции освещенности, вычисленных в узлах равномерной решетки с относительно малым шагом [60]. Такая матрица называется *картой освещенности*. Недостаток использования карт освещенности заключается в том, что их хранение в явном виде, как правило, требует неоправданно больших объемов памяти. Второй способ — вычислять значения функции в узлах некоторой сетки, как правило, треугольной [72]. Это сокращает объем данных и, кроме того, предоставляет довольно простой способ вычисления самих значений в вершинах треугольника: энергия фотона, попавшего в треугольник, делится между вершинами этого треугольника в долях, определяемых степенью удаления фотона от вершин. Недостаток метода в том, что сетка не может учитывать особенностей реальной функции освещенности, поскольку определяется до начала процесса вычисления из некоторых априорных соображений.

Альтернативой такому подходу является построение адаптивной сетки по результатам вычисления карты освещенности. Возможность построения адаптивных сеток для функций освещенности ранее рассматривалась, например в [70]. В этой работе использовался стандартный для адаптивной триангуляции подход [67]: сначала строилась мелкая треугольная сетка, потом из нее последовательно удалялись «лишние» вершины. Кроме того,

процесс вычисления значений функций освещенности и процесс прореживания сетки выполнялись совершенно независимо.

2.3.4 Вычисление значений освещенности

Вычисление значений функции освещенности является примером *задачи оценки плотности* [71]. Задача эта заключается в восстановлении по конечному числу событий закона их распределения (функции плотности распределения).

В данном случае событиями являются следы фотонов, а восстанавливаемая функция освещенности — функцией плотности распределения фотонов.

Существует несколько методов решения задачи оценки плотности.

Проще всего разбить всю область, на которой требуется найти функцию плотности, на равновеликие квадраты (рассматривается двумерный вариант), и значение функции в каждом квадрате определять как количество попаданий событий (в нашем случае фотонов), деленное на его площадь. Строго говоря, результат следует нормировать, поскольку, по определению, функция плотности распределения должна иметь равный 1 интеграл по области определения.

Очевидно, что такой метод дает очень грубое приближение и для рассматриваемой задачи едва ли может быть применим.

Достаточно хороший результат может дать *метод ядер*. Каждому событию ставится в соответствие *ядро* — некоторая быстро затухающая функция, например, финитное приближение гауссиана. Центр носителя ядра помещается в точку события. Функция плотности получается суммированием всех ядер. Фактически ядро является сглаживающим фильтром, который

превращает область влияния фотона из точки в круг. Размер ядра приходится подбирать вручную.

Несколько более сложен *метод ближайших соседей*. Пусть требуется найти значение функции плотности в некоторой точке. Строится круг с центром в этой точке и таким радиусом, чтобы внутри круга оказалось заданное число событий. Значение в точке вычисляется как количество попавших в круг событий, деленное на его площадь. Метод можно усложнить, считая вклад каждого события, оказавшегося в круге, зависящим не только от радиуса круга, но и от расстояния до его центра. Фактически это комбинация метода ближайших соседей и метода ядер, где сглаживающая функция (ядро) выбирается в зависимости от плотности событий вблизи рассматриваемой точки.

Подробнее о реализации различных методов оценки плотности применительно к задаче реконструкции освещенности и сравнении этих методов см. [60].

В [70] с помощью метода ядер рассчитывается карта освещенности. Она триангулируется (для регулярной решетки, которой, собственно, является карта освещенности, эта операция тривиальна), потом с помощью специального алгоритма прореживания удаляются «лишние» узлы, производится ретриангуляция соответствующих областей. Такой процесс, как и любой алгоритм адаптивной триангуляции, выполняется достаточно медленно.

Для построения функции освещенности в виде адаптивной треугольной сетки предлагается разработать метод на основе многомасштабного анализа. Интересна не задача триангуляции уже готовой карты освещенности, а объединение процесса расчета освещенности и построения адаптивной

сетки в один этап.

2.4 Описание метода реконструкции освещенности

2.4.1 Начальное приближение функции освещенности

Прежде всего нужно определить, какой объект будет подвергаться обработке. Функция освещенности — это двумерный сигнал (причем, на практике дискретный). Исходные данные — список попавших на поверхность фотонов — таковым не является. Следовательно, нужно на основе этих данных построить двумерный сигнал, который будет считаться начальным приближением функции освещенности, и к которому будут применяться все дальнейшие преобразования. На поверхности строится решетка с постоянным шагом, определенным пользователем. Каждый узел решетки получает значение — количество фотонов, которые оказались ближе всего к данному узлу. Другой вариант, дающий несколько лучшие результаты, — распределять энергию фотона (ее мы договорились считать 1) между четырьмя ближайшими к нему узлами, например, линейно.

Чем мельче выбранный шаг решетки, тем с бóльшим разрешением можно теоретически восстановить функцию освещенности, однако с уменьшением шага растет уровень шума. Очевидно, что уровень шума снижается при увеличении количества фотонов. Вообще, количество фотонов можно подобрать так, чтобы такой объект достаточно хорошо приближал бы искомую функцию освещенности, (т.е. фактически являлся бы картой освещенности), однако это количество должно быть очень велико (о чем и свидетельствует медленная сходимость метода Монте-Карло).

2.4.2 Структура преобразования

Попытка применить вейвлет-анализ для обработки карт освещенности уже предпринималась автором, результаты работы были изложены в [62]. Изначально предполагалось применять вейвлет-преобразование непосредственно к начальному приближению, описанному выше, и строить адаптивную сетку с помощью дерева узлов. Подавление шума должно было производиться на этапе обратного преобразования за счет обращения в нуль соответствующих вейвлет-коэффициентов. Однако оказалось, что шумопонижающих возможностей вейвлет-преобразования (по соображениям простоты было выбрано преобразование Хаара) оказалось недостаточно, и в результате фазу оценки плотности пришлось сохранить. Таким образом, работа [62] оказалась выполненной по той же схеме, что и [70], только вместо метода ядер для оценки плотности была использована упрощенная версия метода ближайших соседей, а вместо так называемого метода прореживания для построения адаптивной сетки применялся описанный выше метод триангуляции по дереву вейвлет-преобразований.

В дальнейшем был проведен ряд экспериментов с различными вейвлет-базисами, однако удовлетворительные результаты так и не были получены. По-видимому, это связано с тем, что используемая схема обычного двумерного преобразования вообще не является удачной для данного приложения.

Для эффективного сглаживания сильно зашумленного двумерного сигнала требуются двумерные фильтры, которые не получаются композицией одномерных. Существуют схемы преобразований, использующие двумерные фильтры в явном виде (например, «шахматная доска», упоминаемая [49]), однако реализация таких схем, а тем более построение на их основе тре-

угольных сеток, более сложны, чем реализация обычных схем.

Предлагается следующее решение: при прямом преобразовании использовать двумерный НЧ фильтр, но использовать то же разбиение пространства, что и при композиции одномерных преобразований. Это эквивалентно двумерной свертке сигнала с последующем применением оператора прореживания $\downarrow_2[\bullet]$ вдоль каждой из координат.

Возникает вопрос о способе вычисления ВЧ составляющих и о выборе фильтра. ВЧ составляющую предлагается вообще не вычислять, а хранить вместо этого все НЧ представления. Фактически, единственный недостаток такого подхода в том, что при этом происходит увеличение объема хранимой информации. Но в двумерном случае, как нетрудно установить, объем данных увеличится менее чем в полтора раза, кроме того, эти данные нужны только на этапе вычислений, после построения сетки их сохранять не требуется. Выгода такого решения в первую очередь в том, что снимается проблема восстановления сигнала (с разным разрешением он хранится в явном виде) и, следовательно, нет необходимости специально подбирать фильтры так, чтобы они обеспечивали восстановление. Таким образом, выбор фильтра определяется только его сглаживающими характеристиками. Мало того, возможно даже применение разных фильтров на одном уровне разрешения (об это будет сказано ниже). Кроме того, хранение НЧ составляющих в явном виде сокращает объем вычислений приблизительно два раза на этапе прямого преобразования, а обратное преобразование не требуется вообще.

2.4.3 Дерево преобразования и триангуляция

Построение дерева в этом случае практически ничем не отличается от построения дерева узлов, которое рассматривалось в п. 2.2.1, только каждой вершине дерева следует приписывть не вейвлет-коэффициенты, а непосредственно то значение, которое должно быть в соответствующем узле решетки. То есть дерево имеет такой вид, как если бы по нему уже было выполнено обратное преобразование (такое дерево является разновидностью пирамиды гауссианов [24]).

Нужно заметить, что листовым вершинам такого дерева приписаны значения элементов начального приближения, не подвергнутого какой-либо обработке. Эти данные не должны использоваться при формировании окончательного результата, и их следует отсечь, уменьшив таким образом глубину дерева на 1. Однако это приведет к падению разрешения результата (вернее, к падению максимально возможного разрешения данных, по которым будет формироваться результат). Поэтому целесообразно строить начальное приближение с разрешением, на 1 большим, чем, то, которое требуется для представления результата.

Для того, чтобы отсечь нуль-деревья, для нелистовых вершин производится сравнение приписанных к ним значений со значениями, приписанными к четырем вершинам-потомкам. Разности этих значений являются в некотором роде аналогами вейвлет-коэффициентов, но в отличие от «настоящего» вейвлет-преобразования, для каждой нелистовой вершины их оказывается не три, а четыре. Проще всего объявлять вершину корнем нуль-дерева, если максимум модуля разности приписанного к ней значения и значений, приписанных к четырем ее потомкам, не превышает заданного

числа.

Фаза триангуляции остается без изменений, поскольку входными данными для этого этапа является итоговое несбалансированное дерево со значениями в листовых вершинах, и то, как было построено это дерево, не имеет никакого значения.

2.4.4 Выбор фильтров

Рассмотрим вопрос о том, какие фильтры используются для расчетов.

Проще всего взять какой-нибудь один сглаживающий фильтр. Заметим, что даже в этом случае действие фильтра будет более сложным, чем обычная однократная свертка. Действительно, фильтр применяется каждый раз при вычислении следующего уровня разрешения, причем на каждом шаге размер сигнала уменьшается, что эквивалентно растяжению фильтра. То есть, чем ниже уровень разрешения, тем выше степень сглаживания. Такой подход может дать неплохие результаты для плавно меняющихся функций освещенности, не имеющих перепадов: даже фильтры небольшого размера могут обеспечить хорошее подавление шума, для этого достаточно формировать сетку по значениям на низких разрешениях. При наличии же перепадов может произойти следующее: если взять значения высоких уровней, то шумы на этом уровне могут оказаться недостаточно хорошо подавлены, а если на низких, то перепады окажутся сильно размыты. Можно, конечно, для каждой конкретной функции специально подбирать фильтр — фактически это будет соответствовать ручному выбору размера ядра в обычном методе ядер.

Однако возможен и другой метод. Он более сложен, но дает лучшие ре-

зультаты. Фактически это модификация метода переменных ядер на основе ближайших соседей.

На каждом шаге каждый элемент текущего сигнала осредняется по некоторому фильтру, то есть его значение и значения ближайших к нему элементов суммируются с весами, которыми являются элементы фильтра. Сумма коэффициентов фильтра равна 1, что гарантирует сохранение энергии осредненного сигнала. В простейшем случае фильтр для всех элементов один и тот же, и осреднение реализуется обычной сверткой по этому фильтру. Метод ближайших соседей предполагает осреднение по области, в которой оказалось не менее заданного числа фотонов. Вообще говоря, размер такой области может быть в каждом случае свой. Предлагается ограничить выбор фиксированным числом возможных областей, определив несколько фиксированных фильтров разного размера. Выбор фильтра производится для каждого элемента в зависимости от того, какова энергия сигнала (и, следовательно, сколько фотонов) внутри области, задаваемой фильтром. Чтобы узнать эту энергию, надо просто просуммировать (с весом 1) значения всех элементов, попавших в область. Заметим, что вычисление таких сумм для всех точек сигнала эквивалентно свертке этого сигнала по фильтру-маске, полученному обращением в 1 всех ненулевых элементов исходного фильтра. Далее для каждого элемента сигнала выбирается фильтр так, чтобы количество фотонов-соседей, попавших в область, заданную этим фильтром, было бы ближе всего к заданному числу.

Количество фильтров может быть, разумеется, разным. В наших опытах использовались три фильтра, которые условно можно считать «малым», «средним» и «большим». Если фильтров три, то расчет количества

соседей можно удешевить: считать количество соседей не для каждого из них, а только для среднего фильтра. Выбор фильтра производится следующим образом. Если для данного элемента количество соседей (по среднему фильтру) составило менее 2/3 от заданного числа, то осреднение производится большим фильтром, если более 4/3, то малым, в остальных случаях средним.

Ниже приводится пример возможной тройки фильтров.

$$\frac{1}{32}\begin{bmatrix} 1 & 2 & 2 & 1 \\ 2 & 3 & 3 & 2 \\ 2 & 3 & 3 & 2 \\ 1 & 2 & 2 & 1 \end{bmatrix};\quad \frac{1}{124}\begin{bmatrix} 1 & 2 & 3 & 3 & 2 & 1 \\ 2 & 4 & 5 & 5 & 4 & 2 \\ 3 & 5 & 6 & 6 & 5 & 3 \\ 3 & 5 & 6 & 6 & 5 & 3 \\ 2 & 4 & 5 & 5 & 4 & 2 \\ 1 & 2 & 3 & 3 & 2 & 1 \end{bmatrix};\quad \frac{1}{344}\begin{bmatrix} 1 & 2 & 3 & 4 & 4 & 3 & 2 & 1 \\ 2 & 5 & 6 & 7 & 7 & 6 & 5 & 2 \\ 3 & 6 & 8 & 9 & 9 & 8 & 6 & 3 \\ 4 & 7 & 9 & 10 & 10 & 9 & 7 & 4 \\ 4 & 7 & 9 & 10 & 10 & 9 & 7 & 4 \\ 3 & 6 & 8 & 9 & 9 & 8 & 6 & 3 \\ 2 & 5 & 6 & 7 & 7 & 6 & 5 & 2 \\ 1 & 2 & 3 & 4 & 4 & 3 & 2 & 1 \end{bmatrix}.$$

Производились эксперименты и с другими фильтрами, например, константными и гауссовыми разных размеров.

2.4.5 Примеры работы метода

На с. 112-113 показаны некоторые примеры работы метода.

На рис. I.3 слева показано изображение, которое рассматривается как некоторая эталонная функция освещенности. Справа показан результат попадания на поверхность 200 тыс. фотонов в процессе трассировки Монте-Карло.

Эталонное изображение можно считать пределом плотности распределения фотонов для данного шага оцифровки. Пусть исходная функция оци-

фрована на решетке размера 256×256 (именно такого размера изображение на рисунке). Если считать, что один фотон увеличивает яркость точки изображения на 1, а яркость меняется в диапазоне от 0 до 255, то количество фотонов, которое необходимо для получения эталонного изображения, равно суммарной яркости этого изображения. Для данного изображения это $6.3 \cdot 10^6$. Следовательно, 200 тыс. фотонов составляет 3% от предельного числа.

На рис. I.4 изображен результат реконструкции функции по имеющимся данным — слева показана адаптивная треугольная сетка (2377 вершин, 4683 треугольника), справа треугольники сетки залиты по методу Гуро.

Рис. I.5 демонстрирует исходные данные и результат реконструкции той же функции по 500 тыс. фотонов (8% от предельного числа).

2.5 Анализ результатов

В данной главе рассматривается применение многомасштабных методов анализа объектов, параметризуемых на плоскости, для построения их адаптивных сеточных представлений.

Большинство методов упрощения сеточных представлений, включая многомасштабный метод [39] (см. п. 2.2.2), основаны на прореживании исходной сетки, т.е., пользуясь терминологией теории обработки сигналов, осуществляет точечную выборку. Предложенный метод создает новую сетку, вершины которой являются взвешенной выборкой вершин исходной сетки. Это улучшает аппроксимирующие свойства метода, кроме того, появляется возможность на этапе построения сетки выполнять дополнительную обработку сигнала, например, сглаживание.

Метод лишен таких недостатков алгоритмов адаптивной триангуляции (напр., [67]), как сложные критерии выбора удаляемых вершин, необходимость повторной локальной триангуляции, возможность «перехлеста» треугольников после удаления вершины и пр. В отличие от [39], предложенная схема преобразования не содержит «неравноправных» вершин. По производительности метод сравним с [39] и несколько проще его в реализации.

Базовый вариант метода может быть применен для широкого круга задач, например, для описания виртуальных ландшафтов. Он основан на обычном двумерном вейвлет-преобразовании, которое допускает достаточно эффективную реализацию и, после незначительных модификаций алгоритма, позволяет ввести дополнительные операции: автоматическое управление уровнем детализации, прогрессивную передачу объекта и др.

При восстановлении функции освещенности требуется более сложный анализ исходной информации, на что и ориентирован соответствующий вариант. Для его реализации выбрана схема, близкая к построению пирамиды гауссианов [24]. Она требует большего расхода памяти, но снимает практически все ограничения на способы вычисления НЧ составляющих, что является особенно важным при решении данной задачи.

В отличие от других методов построения функции освещенности (напр., [70]), данный подход объединяет в единый процесс фазу собственно восстановления функции и фазу построения для этой функции сеточного представления, используя при этом одну и ту же структуру данных — дерево узлов. Это упрощает процедуру обработки и сокращает общий объем вычислений.

На примере реализации двух вариантов одного метода показано, как,

учитывая особенности конкретной задачи, можно изменять схему анализа информации, представление объекта, построенного на основе результатов такого анализа, и алгоритм дальнейшей обработки этого представления.

Глава 3

Многомасштабное представление линий уровня

3.1 Описание задачи

Задача построения линий уровня (изолиний) постоянно возникает в самых различных областях — научной визуализации, геологии, картографии и т.д.

В определенной степени эту задачу можно считать решенной, как только на основе входных данных (как правило, это прямоугольная таблица значений исследуемого параметра) найдена последовательность точек, через которую нужно провести кривую — эта кривая и будет линией уровня. Задача построения кривой практически любой гладкости по заданной последовательности точек (сплайновой кривой) также решалась множеством различных способов [17, 19, 65]. Сплайны для построения линий уровня используются, например, в графическом пакете ГРАФОР [1].

Однако на практике возникают дополнительные проблемы. Исходные данные, по которым должна быть построена кривая, могут содержать различные искажения. Например, в геологии в качестве исходных данных мо-

гут выступать результаты реальных замеров некоторого параметра, сделанные в «полевых» условиях. Построенные по таким данным линии будут искажены высокочастотным шумом, который желательно подавить.

Построение сплайновой кривой, как правило, состоит из вычисления коэффициентов полинома нужной степени на каждом фрагменте кривой, и последующего вычисления этого полинома для заданных значений параметра. Однако такой метод не эффективен, если кривая состоит из большого числа относительно коротких фрагментов.

В современных операционных системах с оконным интерфейсом размер окна, в котором осуществляется вывод, может быть изменен пользователем одним движением «мыши». Хотелось бы, чтобы изображение в окне могло автоматически перерисовываться с учетом изменения размера, причем время перерисовки должно быть как можно меньше.

Предлагаемый способ представления линий уровня в виде кубических B-сплайновых кривых [17, 19, 65] дает возможность применить к ним B-сплайновый вейвлет-анализ, который позволяет, во-первых, сгладить кривую, а, во-вторых, эффективно отобразить ее с любым разрешением, заданным пользователем. При этом алгоритм отображения настолько прост, что, после определенных доработок, может быть реализован аппаратно.

Описание метода и результаты его реализации опубликованы в [14].

3.2 Построение последовательности управляющих точек

Опишем достаточно известный алгоритм построения последовательности управляющих точек линии уровня, т.е. точек, по которым впоследствии бу-

дет проведена кривая, являющаяся, по сути, искомой линией. (Идеи этого метода легли в основу более сложного алгоритма для построения поверхностей уровня, получившего название «марширующие кубы» [52]).

По условию задачи исходными данными является прямоугольная таблица размера $M_x \times M_y$ значений некоторого параметра z. Всегда можно считать, что это таблица значений в точках с целыми координатами некоторой функции $z(x, y)$, определенной на прямоугольной области $\Pi = [0, M_x - 1] \times [0, M_y - 1]$, то есть

$$z(i, j) = z_{i,j}, \quad i = \overline{0, M_x - 1},\ j = \overline{0, M_y - 1}.$$

Множество точек $(i, j),\ i = \overline{0, M_x - 1},\ j = \overline{0, M_y - 1}$, определяет на Π *сетку* $\mathcal{K}$ (эти точки в таком случае именуются *узлами* сетки). *Ребрами* сетки $\mathcal{K}$ назовем соединяющие узлы отрезки единичной длины. Ребра направлены вдоль оси OX или OY, но не по диагонали. Обозначаются ребра координатами пары узлов, которые они соединяют, например $(i, j)-(i, j+1)$ или $(i, j) - (i + 1, j)$.

Пусть теперь требуется построить на Π линию уровня некоторого значения Z функции $z(x, y)$.

Для каждого ребра сетки значения в узлах, которые ребро соединяет, сравниваются с Z. Если значение в одном из узлов оказалось не меньше Z, а в другом — строго меньше Z, то это значит, что линия уровня пересекает это ребро в некоторой точке. Если же оба значения не меньше (или, наоборот, меньше) Z, то линия уровня через это ребро не проходит.

Если установлено, что линия пересекает ребро, то требуется найти координаты точки пересечения.

Проще всего в качестве такой точки брать середину ребра. Но это мо-

жет оказаться достаточно грубым приближением. Кроме того, если одно и то же ребро пересекают линии разных уровней, то все они будут проходить через одну и ту же точку, что даст нежелательный эффект при отображении. Предлагаемый подход — линейно проинтерполировать функцию $z(x, y)$ вдоль ребра и найти точку, в которой интерполированная функция примет значение Z.

Таким образом, найдены все ребра, которые пересекаются изолинией (будем называть такое ребро *ребром с пересечением*), и для каждого из этих ребер найдены координаты точки пересечения.

Следующим этапом является соединение точек в последовательности.

Перед тем, как описать этот алгоритм заметим следующее.

Любые четыре узла вида (i, j), $(i+1, j)$, $(i, j+1)$ и $(i+1, j+1)$ являются вершинами квадрата, стороны которого образуют четыре ребра, соединяющие соответствующим образом эти узлы.

Каждое некраевое ребро является общей стороной двух таких квадратов, каждое краевое ребро является стороной только одного квадрата.

Теперь опишем алгоритм построения последовательностей.

- **Шаг 0.** Все ребра с пересечениями объявляются *непомеченными*, все ребра без пересечений — *помеченными*.

- **Шаг 1.** Ищется любое непомеченное ребро (заметим, что любое непомеченное ребро — это заведомо ребро с пересечением). Оно помечается, и объявляется *текущим*. Соответствующая точка пересечения инициализирует последовательность. Кроме того, ребро запоминается как условное начало линии. Любой квадрат, стороной которого является ребро, объявляется *текущим*, т.е. квадратом, в котором будет произ-

водиться поиск следующей точки. Если ребро оказалось краевым, то текущий квадрат определяется, естественно, единственным образом.

- **Шаг 2.** Поиск следующей точки. В текущем квадрате ищутся стороны, которые являются непомеченными ребрами. Если поиск пересечений был выполнен правильно, то непомеченных ребер всегда должно быть либо одно, либо три. Исключение составляет случай, когда одно из помеченных ребер (кроме текущего) оказалось условным началом линии. Этот случай будет рассмотрен ниже.

  Если не помечено только одно ребро квадрата, то соответствующая ему точка пересечения добавляется в последовательность, ребро помечается и становится текущим. (Это соответствует тому, что только одна линия пересекает квадрат). Если непомеченных ребер три (это значит, что квадрат пересекают две линии), то в качестве следующей точки можно выбрать точку, соответствующую любому ребру, *не противолежащему* последнему помеченному ребру (в противном случае вторая линия должна будет пересечь первую, что не допускается). Если точка выбрана, то она добавляется в последовательность, соответствующее ребро помечается и становится текущем.

  Если же одно из нетекущих ребер квадрата оказалось началом линии, то непомеченных ребер в квадрате может быть либо ни одного, либо два. В первом случае процесс построения последовательности останавливается, и последовательность объявляется *замкнутой*, т.к. ее конец совпал с началом. Во втором случае, при условии правильно выполненного поиска пересечений, ребро — начало линии никогда не будет противолежащим текущему (всегда будет иметь с ним общий узел).

Поэтому, можно либо продолжить процесс, выбрав следующую точку на непротиволежащем непомеченном ребре, пометив его и сделав текущим, либо остановить процесс, объявив последовательность замкнутой.

- **Шаг 3.** Текущее ребро может оказаться либо краевым, либо нет. Если ребро не краевое, то текущим объявляется тот квадрат, стороной которого является ребро, отличный от того, который только что был текущим. После этого выполняется шаг 2.

  Если ребро краевое, то последовательность объявляется *незамкнутой*. Текущим снова становится ребро — условное начало линии. Если оно краевое, то процесс построения останавливается. В противном случае текущим квадратом становится квадрат, стороной которого является это ребро, но который не был выбран в первый раз. После этого выполняются шаги 2 (причем каждая новая точка добавляется не в конец, а в начало последовательности) и 3 то тех пор, пока текущее ребро не станет краевым. На этом процесс построения останавливается.

- **Шаг 4.** Если остались непомеченные ребра, то снова выполняется шаг 1.

В результате выполнения данного алгоритма получается одна или несколько последовательностей, каждая из которых соответствует одной линии, замкнутой или незамкнутой (т.е. упирающейся в границы области).

3.3 Построение линии

3.3.1 Уточнение формулировки задачи

Теперь рассмотрим вопрос построения линии по заданной последовательности точек $\mathbf{C} = \{\mathbf{c}_j\}_{j=0}^{N-1},\ N \in \mathbf{Z},\ N > 0$, где $\mathbf{c}_j \in \Pi,\ j = \overline{0, N-1}$. Точки последовательности $\mathbf{C}$ будем далее называть *управляющими точками* или *узлами*, саму последовательность $\mathbf{C}$ — *управляющей последовательностью*.

Проанализируем задачу и уточним ее формулировку.

Простейший способ построения кривой — последовательное соединение точек последовательности отрезками прямой, т.е. построение ломаной. Достоинством этого способа является его простота. Однако не менее очевидны серьезные недостатки такого подхода. Интуитивно понятно, что линия уровня должна быть по возможности «плавной». Достаточно неформальному понятию «плавности» в определенной степени соответствует математическое свойство гладкости кривой хотя бы первого порядка. Ломаная таким свойством не обладает (она лишь *кусочно*-гладкая). Можно допустить, что при определенных условиях отображенная ломаная будет выглядеть достаточно «плавной». Для этого, в частности, желательно, чтобы любые три подряд идущие точки $\mathbf{c}_{j-1}$, $\mathbf{c}_j$ и $\mathbf{c}_{j+1}$ последовательности лежали бы почти на одной прямой, т.е. сумма длин отрезков $\overline{\mathbf{c}_{j-1}\,\mathbf{c}_j}$ и $\overline{\mathbf{c}_j\,\mathbf{c}_{j+1}}$ была бы близка к длине отрезка $\overline{\mathbf{c}_{j-1}\,\mathbf{c}_{j+1}}$. При определенном разрешении такая ломаная может выглядеть «плавной». Однако стоит только в несколько раз увеличить выводимый объект, как его кусочно-линейная природа станет очевидной. Таким образом, даже при самом удачном расположении точек в

управляющей последовательности, простое их соединение отрезками дает *плохо масштабируемый* результат.

Более предпочтительный вариант — построить по точкам управляющей последовательности кривую второго или третьего порядка гладкости, т.е. *сплайновую* кривую. Являясь более гладкой, чем ломаная, она значительно лучше масштабируется. Однако и с отображением гладких сплайновых кривых возникает ряд проблем. Выше было отмечено, что при удачном расположении управляющих точек даже ломаная может выглядеть достаточно плавной кривой. Но возможна и прямо противоположная ситуация, когда при неудачном расположении точек даже гладкая кривая, построенная по ним, будет выглядеть совсем не плавной. Такая ситуация особенно вероятна в случае «плохих», т.е. искаженных входных данных, о чем уже говорилось в начале главы. К этому также добавляется погрешность вычисления координат управляющих точек. Таким образом, реально найденные управляющие точки наверняка окажутся смещенными относительно управляющих точек, которые должна была иметь гипотетическая гладкая линия. Построенная по таким точкам линия будет искажена высокочастотным шумом, причем этот шум будет так или иначе проявляться вне зависимости от того, какой гладкости кривая будет построена по управляющей последовательности.

Получается, что перед тем, как вывести кривую, может понадобиться так преобразовать управляющую последовательность, чтобы подавить высокочастотный шум. Такая процедура носит название *сглаживания* кривой.

Таким образом, задача построения линии распадается на три подзадачи.

1. Сглаживание кривой. Модификация последовательности управляющих

точек с целью подавления шумов и искажений.

2. Масштабирование кривой. Модификация последовательности управляющих точек с целью вывода кривой с разной степенью разрешения.

3. Отображение. Вывод на графическое устройство (или в графический файл) кривой по последовательности управляющих точек.

(Под модификацией последовательности точек подразумевается изменение координат точек последовательности, удаление точек, добавление новых).

3.3.2 B-сплайновые кривые и вейвлеты

B-сплайновые кривые [19, 65] вводятся как кривые, составленные из фрагментов — *элементарных B-сплайновых кривых*. Например, кубическая B-сплайновая кривая γ, заданная управляющей последовательностью $\mathbf{C}$, определяется как объединение элементарных кривых следующего вида:

$$\gamma_j(t) = \frac{(1-t)^3}{6}\mathbf{c}_{j-1} + \frac{3t^3-6t^2+4}{6}\mathbf{c}_j + \frac{-3t^3+3t^2+3t-1}{6}\mathbf{c}_{j+1} + \frac{t^3}{6}\mathbf{c}_{j+2}, \quad (3.1)$$

$$t \in [0,1], \quad j = \overline{0, N-2}.$$

Для построения кривых $\gamma_0(t)$ и $\gamma_{N-2}(t)$ требуется ввести две дополнительных точки. Для незамкнутых кривых они вводятся следующим образом:

$$\mathbf{c}_{-1} = (\mathbf{c}_0 - \mathbf{c}_1) + \mathbf{c}_0, \quad \mathbf{c}_N = (\mathbf{c}_{N-1} + \mathbf{c}_{N-2}) + \mathbf{c}_{N-1},$$

и для замкнутых кривых:

$$\mathbf{c}_{-1} = \mathbf{c}_{N-1}, \quad \mathbf{c}_N = \mathbf{c}_0.$$

B-сплайновая кривая является аппроксимирующей, а не интерполирующей, т.е. она не проходит через свои управляющие точки (если только они не лежат на одной прямой). Может показаться, что это не позволяет использовать эти кривые для построения линий уровня, если управляющие точки были найдены с помощью алгоритма, описанного в п. 3.2, поскольку эти точки должны принадлежать кривой. Тем не менее, именно B-сплайновые кривые были выбраны для построения линий уровня.

Выше было сделано предположение, что координаты точек были вычислены с ошибкой, т.е. они смещены относительно соответствующих точек искомой кривой. Величину и направление смещения определить невозможно, их можно считать случайными величинами. Эти случайные величины и порождают высокочастотный шум, искажающий кривую.

B-сплайновая кривая устроена таким образом, что каждая опорная точка влияет на форму кривой, т.е. «тянет» на себя соответствующий фрагмент кривой, однако фрагмент этой точки не достигает. Таким образом, B-сплайновая кривая в определенном смысле гасит разнос точек, и, следовательно, изначально содержит в себе шумопонижающие (сглаживающие) свойства. Следовательно, применение B-сплайновых кривых в данном случае вполне оправдано.

Кроме того, следует заметить, что если управляющая последовательность оказалась «хорошей», т.е. любые три подряд идущие точки лежат «почти на одной прямой», что соответствует отсутствию (или очень низкому уровню) шума (см. п. 3.3.1), то кривая пройдет очень близко к этим точкам, и при построении кривой ошибка практически не будет заметна.

Однако основной аргумент в пользу B-сплайновых кривых — это

возможность применить к ним вейвлет-анализ, а именно, B-сплайновый вейвлет-анализ [25, 73], с помощью которых предлагается решить все упомянутые выше задачи.

Существует иной вариант задания B-сплайновых кривых. Координаты точек последовательности $\mathbf{C}$ рассматриваются как коэффициенты разложения векторнозначной функции $\gamma(t)$ по базису элементарных B-сплайнов, т.е.:

$$\gamma(t) = \sum_{j=0}^{N-1} \mathbf{c}_j \mathcal{B}_j(t), \quad t \in [0,1]. \tag{3.2}$$

При правильном определении функций $\mathcal{B}_j(t)$ кривые, заданные формулами (3.1) и (3.2), совпадают, за исключением, возможно, фрагментов вблизи краевых точек $\mathbf{c}_0$ и $\mathbf{c}_{N-1}$ (построение кривых вблизи краевых точек будет рассмотрено ниже).

Обращаем внимание на то, что изменение любой точки приведет к изменению локального участка кривой, но не кривой в целом. Действительно, согласно формуле (3.1), любая точка $\mathbf{c}_j$ участвует в формировании только четырех фрагментов: $\gamma_{j-2}(t)$, $\gamma_{j-1}(t)$, $\gamma_j(t)$ и $\gamma_{j+1}(t)$, на формирование всех остальных фрагментов кривой она никак не влияет. Локально влияет на форму кривой и каждое слагаемое в формуле (3.2), поскольку элементарные B-сплайновые функции $\mathcal{B}_j(t)$ имеют компактный носитель. Это свойство B-сплайновых кривых позволяет, в частности, утверждать, что любой способ построения кривой вблизи границ не изменит «внутреннюю» часть кривой.

Представление B-сплайновых кривых в форме (3.2) позволяет применить к управляющей последовательности (которая являются дискретным сигналом) B-сплайновое вейвлет-преобразование. При этом полученная на лю-

бом шаге прямого преобразования НЧ составляющая также оказывается управляющей последовательностью некоторой B-сплайновой кривой, являющейся более грубым приближением исходной кривой[1].

Каждый шаг прямого B-сплайнового преобразования (как и любого диадного вейвлет-преобразования) уменьшает количество элементов сигнала (управляющих точек) примерно вдвое (точное число зависит от того, сколько точек было в исходной последовательности и каким образом обрабатываются края кривой). При этом часть высокочастотной информации, естественно, теряется. Таким образом с помощью B-сплайнового вейвлет-преобразования осуществляется *сглаживание* кривой.

Если графическая библиотека и графическое устройство поддерживают работу с B-сплайновыми кривыми, то задачу построения B-сплайновой кривой можно считать решенной, как только найдена соответствующая управляющая последовательность.

Более интересна ситуация, когда нет готового средства построения кривой, и ее приходится строить поточечно. Можно строить в явном виде по формулам (3.1). Недостатки способа очевидны: для каждого звена кривой $\gamma_i(t)$ требуется вычислить коэффициенты соответствующего многочлена третьей степени и выбрать шаг изменения параметра t, затем для каждого текущего значения параметра вычислять значение этого многочлена.

Предлагается другой метод. Допустим, существует относительно недорогой способ изменения исходной последовательности узлов так, чтобы уз-

[1]С точки зрения теории многомасштабного анализа (см. Приложение А) элементарные B-сплайны являются базисными функциями в пространствах B-сплайнов, образующих многомасштабный анализ, т.е. *скейлинг-функциями*. B-сплайновая кривая γ, определенная по формуле (3.2), является элементом одного из пространств $V^{(i)}$ многомасштабного анализа, а точки последовательности $\mathbf{C}$ суть коэффициенты ее разложения по системе скейлинг-функций в этом пространстве.

лов становилось все больше, а построенная по ним кривая не менялась. С увеличением количества узлов управляющая ломаная приближается к самой кривой. Задача, таким образом, сводится к тому, чтобы, учитывая масштаб кривой и разрешение графического устройства, добавлять узлы до тех пор, пока построенная по ним ломаная сама по себе не окажется хорошей аппроксимацией искомой кривой. Преимущество такого метода еще и в том, что он применим не только для растровых, но и для векторных устройств, которые представляют любой объект в виде последовательности отрезков.

Операция по такому изменению последовательности узлов носит название «вставки узла» [17, 65].

Вставка узлов может быть реализована с помощью обратного B-сплайнового вейвлет-преобразования с нулевой ВЧ составляющей. Действительно, шаги прямого вейвлет-преобразования увеличивают количество элементов сигнала (т.е. количество управляющих точек), а нулевая ВЧ составляющая означает, что в описание объекта не добавляется новой информации[2].

Каждый шаг обратного преобразования увеличивает количество точек в последовательности приблизительно вдвое (точное количество зависит от того, сколько точек было в исходной цепочке и от того, как решено обрабатывать края кривой). При этом приблизительно вдвое сокращается расстояние между двумя соседними точками. Таким образом, можно достаточно

[2]Более строго данная операция обосновывается в терминах многомасштабного анализа (см. Приложение А). Увеличение количества управляющих точек без изменения кривой соответствует тому, что ту же самую кривую (элемент подпространства $V^{(i)}$) проецируют на пространство более высокого уровня разрешения $V^{(i+1)}$ без добавления детализирующей информации. Для этого требуется выполнить один шаг обратного B-сплайнового вейвлет-преобразования с ВЧ составляющей, равной нулю.

легко оценить максимально возможный размер отрезка ломаной. Например, при отображении на обычный экран монитора достаточно выполнять преобразование до тех пор, пока максимально возможная длина отрезка ломаной не будет соответствовать 4-5 пикселам. После этого можно просто вывести на экран полученную управляющую ломаную.

Может, однако, возникнуть и обратная ситуация, когда точек изначально очень много, расстояние между ними мало, и при отображении возможно наложение узлов друг на друга. Такое может случиться, если кривую требуется вывести с очень низким разрешением (например, в окне малого размера). В этом случае имеет смысл выполнять прямое преобразование (то есть то же, что делалось для сглаживания) до тех пор, пока отрезки не увеличатся до тех же 4-5 пикселов.

Таким образом, один и тот же инструмент — B-сплайновое вейвлет-преобразование — оказался применим для решения всех трех подзадач: и сглаживания, и масштабирования, и отображения.

Теперь перейдем к более формальному описанию алгоритма.

3.3.3 Реализация вейвлет-преобразования

В гл. 1 было показано, что биортогональное одномерное диадное вейвлет-преобразование полностью определяется четырьмя фильтрами: НЧ анализа $\tilde{\mathbf{h}} = \{\tilde{h}_k\}_{k\in\mathbf{Z}}$, ВЧ анализа $\tilde{\mathbf{g}} = \{\tilde{g}_l\}_{l\in\mathbf{Z}}$, НЧ синтеза $\mathbf{h} = \{h_m\}_{m\in\mathbf{Z}}$ и ВЧ синтеза $\mathbf{g} = \{g_n\}_{n\in\mathbf{Z}}$. Прямое и обратное преобразования реализуются формулами (1.5) и (1.6).

Кубическим B-сплайновым вейвлетам соответствуют следующие коэф-

фициенты фильтров:

$$\begin{aligned}
&\tilde{h}_0 = \tfrac{5}{4},\ \ \tilde{h}_{-1} = \tilde{h}_1 = \tfrac{5}{32},\quad \tilde{h}_{-2} = \tilde{h}_2 = -\tfrac{3}{8},\ \ \tilde{h}_{-3} = \tilde{h}_3 = \tfrac{3}{32};\\
&\tilde{g}_0 = \tfrac{3}{8},\ \ \tilde{g}_{-1} = \tilde{g}_1 = -\tfrac{1}{4},\quad \tilde{g}_{-2} = \tilde{g}_2 = \tfrac{1}{16};\\
&h_0 = \tfrac{3}{4},\ \ h_{-1} = h_1 = \tfrac{1}{2},\qquad h_{-2} = h_2 = \tfrac{1}{8};\\
&g_0 = \tfrac{5}{2},\ \ g_{-1} = g_1 = -\tfrac{5}{16},\ \ g_{-2} = g_2 = -\tfrac{3}{4},\quad g_{-3} = g_3 = -\tfrac{3}{16}.
\end{aligned} \tag{3.3}$$

В рассматриваемой задаче никак не используются ВЧ составляющие. Действительно, при прямом преобразовании используется только НЧ составляющая сигнала, при обратном преобразовании никакой дополнительной информации не добавляется, т.е. ВЧ составляющая тождественно равна нулю. Естественно, что по сигналу $\mathbf{v}^{(i-1)}$ и нулевой ВЧ составляющей исходный сигнал $\mathbf{v}^{(i)}$ не может быть полностью восстановлен, вместо этого получается новый сигнал $\hat{\mathbf{v}}^{(i)}$. Таким образом, при прямом преобразовании используется только первая формула из (1.5)

$$\mathbf{v}_i = \downarrow_2 \left[\mathbf{v}_{i+1} * \tilde{\mathbf{h}}\right], i \in \mathbf{Z}, \tag{3.4}$$

а для обратного преобразования достаточно только первого слагаемого формулы (1.6):

$$\hat{\mathbf{v}}_i = \uparrow_2 [\hat{\mathbf{v}}_i] * \mathbf{h} \quad i \in \mathbf{Z}. \tag{3.5}$$

Также очевидно, что для реализации формул (3.4) и (3.5) требуется знать только два НЧ фильтра $\tilde{\mathbf{h}}$ и $\mathbf{h}$.

3.3.4 Преобразование управляющей последовательности

Исходным сигналом является управляющая последовательность $\mathbf{C}$, которая, в отличие от общего случая, всегда конечна. Подробно вопрос об обработке граничных узлов будет рассмотрен ниже.

Исходному сигналу поставим в соответствие некоторое произвольное значение уровня разрешения i, например $i = 0$. Таким образом, исходный сигнал получит обозначение $\mathbf{v}^{(0)}$. Пусть элементами этого сигнала являются непосредственно точки последовательности $\mathbf{C}$, т.е.

$$\mathbf{v}_0 = \mathbf{C} \equiv \{\mathbf{c}_j\}_{j=0}^{N-1}.$$

Далее для простоты изложения мы будем отождествлять сигналы $\mathbf{v}^{(i)}$ и $\hat{\mathbf{v}}^{(i)}$ с управляющими последовательностями.

Теперь рассмотрим правила обработки граничных узлов. Как было отмечено выше, управляющая последовательность $\mathbf{C}$ может задавать как замкнутую, так и незамкнутую кривую. В обоих случаях предлагается пойти по пути доопределения сигнала за его границами.

Начнем со случая замкнутой кривой.

Очевидно, что продолжение сигнала в этом случае должно быть периодическим. Действительно, если кривая замкнута, значит управляющая ломаная тоже замкнута. Следовательно, сигнал $\mathbf{v}^{(0)}$ можно доопределить так: $v_N^{(0)} := v_0^{(0)},\ v_{N+1}^{(0)} := v_1^{(0)}, \ldots$ и $v_{-1}^{(0)} := v_{N-1}^{(0)},\ v_{-2}^{(0)} := v_{N-2}^{(0)}, \ldots$ Аналогичным образом доопределяется и любой сигнал $\mathbf{v}^{(i)},\ i \in \mathbf{Z}$, когда к нему требуется применить преобразование.

Теперь рассмотрим вопрос о том, сколько элементов должно остаться у сигнала $\mathbf{v}^{(i-1)}$, если у сигнала $\mathbf{v}^{(i)}$ их N_i штук.

Один шаг любого диадного преобразования ставит в соответствие двум коэффициентам $v_{2j}^{(i)}$ и $v_{2j+1}^{(i)}$ один коэффициент $v_j^{(i-1)}$ (и один коэффициент $w_j^{(i-1)}$), что справедливо для любых целых i и j.

Допустим теперь, что число N_i четное. Нетрудно заметить, что в этом случае паре $v_0^{(i)}$ и $v_1^{(i)}$ будет соответствовать элемент $v_0^{(i-1)}$, паре $v_2^{(i)}$ и $v_3^{(i)}$ —

элемент $v_1^{(i-1)}$ и т.д., а паре $v_{N_i-2}^{(i)}$ и $v_{N_i-1}^{(i)}$, будет соответствовать элемент $v_{N_i/2-1}^{(i-1)}$. Следующая пара коэффициентов в силу периодического продолжения совпадет с парой $v_0^{(i)}$ и $v_1^{(i)}$, и соответствующий им коэффициент $v_{N_i/2}^{(i-1)}$ совпадет с $v_0^{(i-1)}$. Таким образом, сигнал $\mathbf{v}^{(i-1)}$ в этом случае полностью определяется $N_i/2$ элементами с индексами от 0 до $N_i/2-1$.

Сложнее дело обстоит в случае, если N_i оказалось нечетным. Элемент $v_{\lfloor N_i/2\rfloor-1}^{(i-1)}$ соответствует паре $v_{N_i-3}^{(i)}$ и $v_{N_i-2}^{(i)}$, а следующий элемент $v_{\lfloor N_i/2\rfloor}^{(i-1)}$ соответствует паре $v_{N_i-1}^{(i)}$ и $v_0^{(i)}$, т.е. совпадения не происходит. С точки зрения сплайновой кривой такая ситуация означает, что две управляющие точки, определяемые элементам $v_0^{(i-1)}$ и $v_{\lfloor N_i/2\rfloor}^{(i-1)}$, соответствуют двум несовпадающим, но пересекающимся участкам кривой.

Простейший способ разрешить эту ситуацию — просто не включать один из конфликтующих элементов (например, с индексом $\lfloor N_i/2\rfloor$) в формируемый сигнал. Этот вариант удобен тем, что не требует отдельно рассматривать случаи четного и нечетного количества элементов. Для любого N_i будет справедливо соотношение $N_{i-1} = \lfloor N_i/2\rfloor$, и сигнал $\mathbf{v}^{(i-1)}$ будет состоять из элементов $v_0^{(i-1)}$, $v_1^{(i-1)}$, …, $v_{N_{i-1}}^{(i-1)}$.

Неплохой результат дает другой вариант. Вместо сигнала $\mathbf{v}^{(i-1)}$ формируется модифицированный сигнал $\bar{\mathbf{v}}^{(i-1)}$ такой же длины $N_{i-1} = \lfloor N_i/2\rfloor$. Все элементы, кроме первого (с индексом 0) совпадают с элементами сигнала $\mathbf{v}^{(i-1)}$, а элемент $\bar{v}_0^{(i-1)}$ является средним арифметическим двух конфликтующих элементов, т.е.

$$\bar{v}_0^{(i-1)} = \frac{1}{2}\left(v_0^{(i-1)} + v_{N_{i-1}}^{(i-1)}\right).$$

Заметим, что формально этот вариант также подходит для случая четного N_i. Действительно в этом случае $v_0^{(i-1)} = v_{N_{i-1}}^{(i-1)}$, следовательно $\bar{v}_0^{(i-1)} = v_0^{(i-1)}$,

и $\bar{\mathbf{v}}^{(i-1)}$ просто совадает с $\mathbf{v}^{(i-1)}$.

Очевидно, что при обратном преобразовании проблем с четными и нечетными длинами сигналов не возникает. Длина сигнала на каждом шаге удваивается, то есть будет заведомо четной. Это, в частности, значит, что длина сигнала $\hat{\mathbf{v}}^{(i)}$ может оказаться меньше, чем длина $\mathbf{v}^{(i)}$ до преобразования. Это вполне допустимо, учитывая то, что полное восстановление сигнала путем обратного преобразования в данной задаче не требуется.

Теперь перейдем к рассмотрению незамкнутой кривой. Строго говоря, это особый объект, который отличается от замкнутых кривых. Предполагается, что незамкнутая B-сплайновая кривая должна начинаться в первой управляющей точке и заканчиваться в последней (напоминаем, что в общем случае управляющие точки кривой не принадлежат). Дополнительно требуется, чтобы в начальной и конечной точке отрезки управляющей ломаной, соединяющие первый узел со вторым и последний с предпоследним, были бы отрезками касательных к кривой.

Существует специальный вид B-сплайновых преобразований, с помощью которых обрабатываются и строятся подобные кривые [34, 73]. Но они требуют введения дополнительных фильтров, что усложняет алгоритм и снижает его эффективность. Для данной задачи такое усложнение едва ли оправдано, поскольку быстрое преобразование всей последовательности (которая может содержать по нескольку сотен управляющих точек) более важно, чем особо тщательная ее обработка вблизи границ.

Предлагается следующий способ преобразования незамкнутых кривых. Он дает удовлетворительный результат, практически не усложняя вычислений.

Для доопределения сигнала используется константное продолжение, то есть для любого уровня разрешения $i \in \mathbf{Z}$ недостающие элементы сигнала определяются так: $v_j^{(i)} := v_0^{(i)},\ j < 0,$ и $v_j^{(i)} := v_{N_i-1}^{(i)},\ j \geq N_i$.

Элементы низкочастотной составляющей $\mathbf{v}^{(i-1)}$ определяются следующим образом: $v_0^{(i-1)} := v_0^{(i)}$, $v_{\lfloor N_i/2 \rfloor}^{(i-1)} := v_{N_i-1}^{(i)}$, элементы с индексами от 1 до $\lfloor N_i/2 \rfloor - 1$ вычисляются по формуле (3.4). Таким образом, сигнал $\mathbf{v}^{(i-1)}$ содержит $N_{i-1} = \lfloor N_i/2 \rfloor + 1$ элементов, причем первый и последний элементы совпадают с первым и последним элементами исходного сигнала $\mathbf{v}^{(i)}$.

При обратном преобразовании коэффициенты определяются следующим образом: $\hat{v}_0^{(i)} := v_0^{(i-1)}$, $\hat{v}_{2N_{i-1}}^{(i)} := v_{N_{i-1}-1}^{(i-1)}$, элементы с индексами от 1 до $2N_{i-1} - 1$ вычисляются по формуле (3.5). Таким образом, сигнал $\hat{\mathbf{v}}^{(i)}$ содержит $2N_{i-1} + 1$ элементов (всегда нечетное количество), причем первый и последний элементы не меняются.

Таким образом, показано, как с помощью прямого и обратного вейвлет-преобразования представить кривую с любым разрешением i (напомним, что для исходной последовательности было решено положить $i = 0$). При этом, на каждом шаге прямого преобразования часть ВЧ информации теряется, количество управляющих точек уменьшается. На каждом шаге обратного преобразования количество управляющих точек увеличивается, а сама кривая при этом практически не меняется.

3.3.5 Сглаживание кривой

Выше отмечалось, что B-сплайновая кривая уже в определенной степени является сглаживающей по отношению к своей управляющей последовательности, однако на практике может потребоваться и более существенное

сглаживание. Для этого выполняется фиксированное (заданное пользователем) количество шагов прямого преобразования $i_0 > 0$. Полученный сигнал $\mathbf{v}^{(-i_0)}$ содержит опорные точки сглаженной кривой.

На практике может возникнуть необходимость использовать фильтр, обладающий лучшими сглаживающими свойствами, чем фильтр $\tilde{\mathbf{h}}$ из (3.3). например, фильтр со следующими коэффициентами (этот фильтр также соответствует определенному виду вейвлет-преобразований):

$$\begin{aligned} &\tilde{h}_0 = \tilde{h}_1 = 0.7031; \qquad \tilde{h}_{-1} = \tilde{h}_2 = -0.1094; \\ &\tilde{h}_{-2} = \tilde{h}_3 = -0.1406; \quad \tilde{h}_{-3} = \tilde{h}_4 = 0.0469. \end{aligned} \tag{3.6}$$

Этот фильтр (назовем его дополнительным) показал неплохие результаты при сглаживании сильно искаженных сигналов. В то же время, если уровень искажений не высок, то имеет смысл применять исходный фильтр $\tilde{\mathbf{h}}$ из (3.3) (будем называть его основным).

3.3.6 Масштабирование и отображение кривой

Рассмотрим процесс отображения кривой по имеющейся управляющей последовательности. Предлагается добавлять в управляющую последовательность точки так, чтобы сама кривая не менялась. При этом управляющая ломаная приближается к кривой. Добавление точек следует производить до того момента, когда вместо кривой можно будет отобразить управляющую ломаную.

Именно на этом этапе кривая адаптируется к текущим параметрам вывода (предыдущие вычисления их никак не учитывали).

Пусть требуется отобразить кривую в окне размера $L_x \times L_y$ пикселов. Вспомним, что управляющая последовательность строилась по сетке разме-

ра $M_x \times M_y$. Если эту сетку отобразить на окно, то расстояние между двумя соседними узлами сетки не будет превышать $d = \max(M_x/L_x,\, M_y/L_y)$ пикселов[3] (это число можно округлить до ближайшего целого, но оно может оставаться и дробным). Очевидно, что размер в пикселах любого звена управляющей ломаной, построенной по этой сетке, также не будет превышать d. Один шаг прямого преобразования, примененного к управляющей последовательности, увеличивает это расстояние приблизительно вдвое, шаг обратного преобразования уменьшает это расстояние также приблизительно вдвое.

Таким образом, считая, что d соответствует уровню разрешения $i = 0$, приблизительный пиксельный размер звена для каждого уровня $i \in \mathbf{Z}$ можно вычислить по формуле:

$$d_i = 2^{-i} d, \quad i \in \mathbf{Z}.$$

В частности, сглаженной кривой, заданной последовательностью $\mathbf{v}^{(-i_0)}$ соответствует число $d_{-i_0} = 2^{i_0} d$.

Пусть теперь задано число d_{max}, определяющее максимально допустимый размер звена, при котором ломаная с нужным качеством визуально приближает кривую. Как правило, для растрового дисплея берется число d_{max} равное 4-5 пикселам.

Теперь не составляет труда найти такой уровень разрешения $i = i_1$, что $d_{i_1} \le d_{max}$, а $d_{i_1-1} > d_{max}$. Управляющая ломаная, построенная по сигналу (последовательности узлов) $\hat{\mathbf{v}}^{(i_1)}$ окажется той самой ломаной, которую можно отобразить вместо соответствующей кривой.

[3]Очевидно, что под расстоянием в данном случае понимается не математическая длина отрезка, а количество пикселов, необходимых для построения этого отрезка без 1, т.е. максимум модуля разности координат пикселов-концов.

Как правило оказывается, что для сглаженной кривой, которой соответствует сигнал $\mathbf{v}^{(-i_0)}$, число $d_{-i_0} > d_{max}$, т.е. $-i_0 < i_1$. Следовательно, чтобы получить сигнал $\hat{\mathbf{v}}^{(i_1)}$, требуется применить к сигналу $\mathbf{v}^{(-i_0)}$ $i_0 + i_1$ шагов обратного преобразования.

Может, однако, оказаться, что $d_{-i_0} < d_{max}$ и $-i_0 > i_1$ (например, было сильно уменьшено окно, в котором требуется отобразить кривую). Очевидно, что в этом случае можно применить к $\mathbf{v}^{(-i_0)}$ $-(i_0 + i_1)$ шагов прямого преобразования. Однако в этом случае требуется не столько сглаживание сигнала, сколько уменьшение объема данных для его представления, следовательно, рекомендуется использовать только основной фильтр, а не дополнительный сглаживающий.

Очевидно, что если $-i_0$ оказалось равным i_1, то дополнительных преобразований выполнять не нужно.

Отдельно можно упомянуть случай $d_{max} = 1$. Это значит, что размер звена ломаной не превышает одного пиксела, то есть вместо ломаной достаточно отобразить все ее узлы, что соответствует поточечному отображению кривой. Очевидно, что вычислительные затраты и объем памяти для хранения данных в этом случае вырастут, а скорость вывода заметно снизится по сравнению с отображением ломаной. Однако, во-первых, этот случай соответствует максимальному разрешению, с которым можно отобразить кривую при данных параметрах графического устройства. Во-вторых, поточечный вывод позволяет применить подавление лестничного эффекта при построении кривой, если такая функция не поддержана графическим устройством; правда, при этом скорость вывода снизится еще существеннее.

3.4 Реализация и анализ результатов

Данный метод построения и отображения линий уровня был реализован в приложении, работающем в операционной системе MS Windows, без использования каких-либо дополнительных графических библиотек, кроме стандартных функций графического вывода, поддерживаемых этой системой (отобразить пиксел, определить цвет отображенного пиксела, отобразить отрезок).

Было реализовано построение управляющей последовательности, сглаживание, отображение линии с помощью управляющей ломаной, а также поточечное отображение с подавлением лестничного эффекта.

К приложению предъявлялись следующие требования: пользователь должен иметь возможность менять степень сглаживания; построенные линии уровня должны отображаться на экране в окне произвольного размера, и при изменении размера окна пользователем линии должны перерисовываться согласно новым размерам; кроме того, пользователь должен иметь возможность записать изображение в графический файл формата BMP, размер изображения при этом определяет сам пользователь.

На с. 114-115 представлены некоторые результаты работы метода. На рис. II.1 изображены линии, построенные по исходным данным (размер сетки 100×100) без дополнительной обработки. Для демонстрации возможностей сглаживания намеренно взяты данные, дающие сильно зашумленный результат. На рис. II.2 и II.3 показаны линии после двух шагов сглаживания основным и дополнительным фильтрами соответственно. Все линии на рис. II.1-II.3 представлены своими управляющими ломаными (размер звена не превышает 4 пиксела). На рис. II.4 изображены те же линии, что и

на рис. II.3, но в разных масштабах (как уменьшенные, так и фрагмент увеличенного изображения); линии построены поточечно, с подавлением лестничного эффекта.

Итак, описанный выше способ позволяет не просто строить линии уровня таблично заданной функции, но и осуществлять дополнительную обработку (сглаживание) этих линий. Кроме того, он обеспечивает выбор точек, реально необходимых для отображения линии при заданных параметрах области отображения.

Важная особенность предложенного метода — использование одного представления и одного аппарата обработки (вейвлет-анализа) для решения всех поставленных подзадач обработки объекта. Это обеспечивает не только эффективную реализацию метода, но и позволяет расширить набор возможных операций (например, ввести локальную обработку произвольного фрагмента кривой), внося в алгоритм минимум дополнений.

Метод не требует больших вычислительных затрат. Вычисления легко распараллеливаются: прежде всего, каждая кривая обрабатывается независимо от другой, кроме того, вычисление вейвлет-преобразований может выполняться параллельно для разных фрагментов кривой (прямое следствие пространственной локализации вейвлетов). Возможна аппаратная реализация метода (что актуально для фазы отображения).

Глава 4

Генерация текстур

4.1 Постановка задачи

В компьютерной графике к текстурам относят как правило два класса объектов. Первый класс — это обычные «содержательные» изображения, которые наносятся на геометрические объекты. Например, это может быть плоское изображение (фотография) архитектурного сооружения, которое наносится на геометрическую модель того же сооружения.

Второй класс в большей степени соответствует общепринятому понятию текстуры. Это изображение некоторого материала (дерево, мрамор, камень и пр.), рисунка ткани (само латинское слово “textura” означает ткань, строение) и т.д., а также просто некоторый абстрактный узор.

Такие текстуры, как правило, хранятся в виде образцов — небольших изображений. Возникает задача — как по этому образцу создать текстуру, которую можно было бы нанести на объект произвольного размера.

Простое размножение образца по поверхности объекта приведет к ярко выраженному периодическому эффекту, который нежелателен. Кроме того, образец в этом случае должен быть подобран так, чтобы он без видимых

разрывов мог быть состыкован сам с собой с каждой стороны.

Существует несколько *стохастических моделей* для представления подобных текстур [21, 41]. В основе этих моделей лежит гипотеза о том, что образцы текстур — это выборка из некоторого вероятностного распределения. Если, анализируя образец, удастся найти это распределение, то с его помощью можно будет генерировать текстуру любого размера. При этом образец и новая текстура не обязаны ни совпадать, ни содержаться друг в друге, однако должны восприниматься как фрагменты одного и того же объекта (поскольку имеют одинаковое распределение). Текстуры, к которым имеет отношение подобная гипотеза, также принято называть *стохастическими.*

В [41] кроме образца текстуры в качестве входных данных требуется также предоставить образец некоторого шума. Итерационный алгоритм последовательно преобразует оба изображения к результату— текстуре заданного размера.

В [21] делается предположение, что в стохастической текстуре можно выделить две компоненты — детерминированную и случайную. Образец подвергается иерархическому анализу, который по своей структуре приближается к вейвлет-анализу. Далее считается, что определенные уровни полученного представления соответствуют случайной компоненте, а другие — детерминированной. Грубо говоря, это значит, что информацию на определенных уровнях можно некоторым образом «перемешивать» (случайная компонента), а на других — нет (детерминированная компонента).

Эти алгоритмы достаточно сложны как в реализации, так и в смысле объема вычислений. В то же время актуальна задача нанесения текстур в

режиме реального времени, например, при быстрой генерации трехмерных сцен в играх, тренажерах-симуляторах и пр. При этом собственно нанесение текстуры желательно реализовать аппаратно. Вышеупомянутые алгоритмы для таких задач неприменимы. Приходится заранее генерировать текстуры нужного размера, а потом передавать готовые изображения на вход графической карты. В этом случае стохастическая природа текстур уже не играет никакой роли, они обрабатываются как обычные изображения. Недостаток подхода в том, что хранение текстуры в явном виде (даже с учетом сжатия) требует довольно больших объемов памяти (алгоритмы сжатия текстур обеспечивают сжатие не более чем в 6-8 раз[1]). Другой вариант — вообще не пытаться генерировать текстуру на основе образца, а передавать графической карте образец без каких-либо дополнительных преобразований, но, как было отмечено выше, в силу малого размера образца едва ли удастся избежать заметного периодического эффекта.

Хотелось бы получить такое представление текстуры, которое, во-первых, учитывало бы ее стохастическую природу (что позволит сделать представление более компактным, чем сжатое изображение сгенерированной заранее текстуры), а, во-вторых, допускало бы простую и эффективную генерацию (включая возможность аппаратной реализации алгоритма).

Нужно заметить, что существует ряд дополнительных требований к алгоритму генерации текстур в реальном времени (и, соответственно, ко входным данным этого алгоритма), несоблюдение которых существенно снижа-

[1]Хотя алгоритмы сжатия изображений (например, известный JPEG) способны сжимать в десятки и сотни раз, они не применимы для сжатия текстур, поскольку сжатое с их помощью изображение практически невозможно обрабатывать, не произведя распаковки. Алгоритмы сжатия текстур обеспечивают существенно более низкую степень сжатия, поскольку должны удовлетворять целому ряду условий и ограничений (подробнее см., напр., [63]).

ет его ценность. Прежде всего это *локальная генерация*, то есть возможность вычисления любого фрагмента (вплоть до отдельного пиксела) текстуры без необходимости генерации изображения в целом. Важным требованием также является *масштабируемость* текстуры, то есть возможность генерации с разным уровнем разрешения.

Таким образом, формулируется следующая задача. Требуется построить модель стохастической текстуры, которая удовлетворяла бы следующим требованиям:

- Представление текстуры должно быть существенно более компактным, чем изображение той же текстуры в явном виде, сжатое с помощью какого-либо алгоритма сжатия текстур.
- Представление должно допускать локальную генерацию.
- Легко реализуется масштабирование текстуры.
- Алгоритм генерации текстуры по ее представлению должен работать в реальном времени и допускать аппаратную реализацию.

4.2 Описание модели

В основе модели лежит идея о «детерминированно-случайной» природе стохастической текстуры (по аналогии с [21]). Обе компоненты такой текстуры предлагается задавать в явном виде: детерминированную компоненту в виде изображения, а случайную — в виде распределения.

4.2.1 Базовый элемент и репликации

Введем несколько понятий, которые будут использоваться при описании модели.

Базовым элементом называется некоторое произвольное растровое изображение.

Репликацией назовем одну копию (возможно, масштабированную и преобразованную перечисленными ниже способами) базового элемента, которая помещается в выходное изображение.

Будем говорить, что репликация помещена в некоторую точку выходного изображения, если в этой точке окажется начало, т.е. левый верхний угол репликации.

Точку выходного изображения, в которую с отличной от нуля вероятностью может быть помещена репликация назовем *точкой репликации.*

Предположим, что базовый элемент — квадратное растровое изображение с размером стороны $N = 2^I$ точек. Тогда масштабированные версии базового элемента — это квадратные растровые изображения с размерами сторон $2^i,\ i = \overline{1,\ I}$ точек. Индекс i назовем *уровнем разрешения* или просто *разрешением.*

Координаты точек репликации одного уровня разрешения кратны половине размера стороны соответствующих репликаций. Это значит, что на уровне разрешения i, которому соответствует размер стороны репликации 2^i, точки репликации имеют на выходном изображении координаты $\left(2^{i-1}j,\ 2^{i-1}k\right),\ j,\ k \in \mathbf{Z}$ (для простоты изложения будем считать выходное изображение бесконечным).

В каждой точке репликации могут иметь место следующие *события*:

- Базовый элемент может быть наложен без инверсии (*позитивная репликация*), с инверсией (*негативная репликация*), и может быть вообще не наложен (*нет репликации*).

- Базовый элемент может быть *повернут* на 90°, 180° и 270°, либо не повернут (т.е. повернут на 0°).

- Базовый элемент может быть *отражен* или не отражен относительно одной из осей координат (вместо отражения можно также применить *транспонирование* в том смысле, как понимается транспонирование матрицы).

Разумеется, возможны комбинации событий, например, негативная репликация с поворотом на 90° без отражения.

Для каждой конкретной модели определяется вероятность каждого из этих событий в точках репликации.

4.2.2 Построение изображения из репликаций

Предположим, что базовый элемент может иметь точки как с положительной, так и с отрицательной интенсивностью. Точка с нулевой интенсивностью считается прозрачной. Нулевую интенсивность имеет фон базового элемента.

На первом шаге выходное изображение — это прямоугольник заданного пользователем размера с нулевой интенсивностью.

Далее в изображение добавляются репликации. Операция добавления может быть выражена обычным сложением интенсивностей точек формируемого изображения и репликации, но допустимы и другие операции.

Введем обозначения: a — текущая интенсивность некоторой точки формируемого изображения, b — интенсивность репликации в данной точке, $\tilde{a}$ — новая интенсивность точки, полученная после репликации. Тогда возможные операции можно определить так:

- обычное сложение

$$\tilde{a} = a + b; \tag{4.1}$$

- ненулевое наложение

$$\tilde{a} = \begin{cases} a, & b = 0 \\ b, & b \neq 0 \end{cases}; \tag{4.2}$$

- наложение максимума

$$\tilde{a} = \begin{cases} a, & |a| \geq |b| \\ b, & |a| < |b| \end{cases}. \tag{4.3}$$

Две последние операции не линейны и не коммутативны, следовательно, их результат зависит от порядка добавления репликаций в изображение. С помощью этих операций возможно управлять «прозрачностью» репликаций.

Вероятности применения каждой из этих операций также могут быть определены для конкретной модели.

Репликации одного уровня разрешения формируют *слой* выходного изображения. Каждому слою может соответствовать один *весовой коэффициент*. В этом случае все репликации слоя должны умножаться на этот коэффициент. С помощью весовых коэффициентов достигается управление контрастом слоя и, следовательно, степенью его значимости в формируемом изображении. Следует заметить, что понятия слоя и уровня разрешения тесно взаимосвязаны, однако не эквивалентны, о чем будет сказано ниже.

Порядок добавления репликаций может быть произвольным. Возможно, например, послойное формирование изображения. При этом можно указывать, следует ли формировать изображения от верхнего слоя к нижнему, или наоборот. Порядок формирования не имеет значения, если в в модели для добавления репликаций используется только операция сложения (4.1), однако, при использовании некоммутативных операций (4.2) и (4.3) порядок формирования может существенно повлиять на результат.

Последним шагом является добавление к каждой точке изображения фиксированного значения интенсивности фона. Собственно, инициализировать изображение значением интенсивности фона можно было на самом первом шаге, однако в этом случае пришлось бы переопределять операции (4.2) и (4.3), которые используют текущее значение интенсивностей точек формируемого изображения в предположении, что фон изображения, как и фон каждой репликации прозрачный (имеет нулевую интенсивность).

4.2.3 Определение параметров модели

Для задания конкретной модели требуется определить количество слоев и таблицы вероятностного распределения событий в точках репликации каждого слоя (*таблицы событий*) и вероятностное распределение событий в точках репликации каждого слоя.

В простейшем случае одно и то же распределение определяется для всех точек репликации слоя. Другой крайностью является задание вероятностей событий для каждой точки репликации в отдельности, но это лишено практического смысла.

Предлагается задавать распределения для группы точек репликации

внутри прямоугольной или квадратной обрасти слоя. Размер области определяет пользователь. Таким образом, формируется прямоугольная или квадратная таблица, каждая ячейка которой соответствует одной точке репликации. Такая таблица называется *таблицей событий*. Генерация слоя любого размера обеспечивается простым «размножением» таблицы событий по вертикали и горизонтали в требуемом количестве.

Базовый элемент может либо включаться, либо не включаться в описание модели. Последнее означает, что модель предназначена для работы с различными базовыми элементами.

При описании модели также следует указать интенсивность фона всего изображения, весовые коэффициенты каждого слоя и порядок послойного формирования изображения (сверху-вниз или снизу-вверх).

4.2.4 Масштабирование

Предлагаемая модель обеспечивает очень простой способ формирования изображений в масштабе 1:2, 1:4, 1:8 и т.д. Действительно, по умолчанию верхний слой (слой 0) соответствует разрешению I базового элемента, слой 1 — разрешению $I - 1$ и т.д. Если сдвинуть это соответствие (например, сопоставить слой 0 с разрешением $I - 1$, слой 1 с разрешением $I - 2$ и т.д.), то получится в 2 раза уменьшенное изображение. Также возможно и увеличение (например, слою 0 соответствует уровень $I + 1$), но для этого базовый элемент нужно уметь растягивать, до размера, соответствующего уровням, бóльшим, чем I.

4.3 Примеры

«Произвольное вращение»

Простейшая модель называется «произвольное вращение». Таблица событий имеет размер 2×2, которая состоит из двух, расположенных в шахматном порядке «нулевых» ячеек (которым соответствует событие «нет репликации» с вероятностью 1) и двух оставшихся ячеек, в которых задается равномерное распределение угла вращения базового элемента (таблица 4.1). Таблица одинакова для всех слоев. Количество слоев может быть любым, но обычно 3 или 4. Весовые коэффициенты могут быть разными, но в простейшем случае это 1.0 для всех слоев.

Базовый элемент не включен в описание модели. Данная модель может быть использована для генерации новых текстур. Пользователю предлагается самому создать базовый элемент (что можно сделать с помощью любого графического редактора) и, включив его в заготовленную модель, получить результат.

Вместо негативной репликации для данной модели дополнительно предлагается следующее преобразование базового элемента: он складывается со своей инвертированной и повернутой на 180° копией (рис. 4.1). Средняя интенсивность преобразованного элемента всегда равна 0. Интенсивность фона выходного изображения задается, как правило, равной половине максимально интенсивности (т.е. либо 0.5, если интенсивность задается числом с плавающей точкой в диапазоне [0, 1], либо 127, если интенсивность задается целым числом в диапазоне от 0 до 255).

Примеры текстур, полученных с помощью модели «произвольное враще-

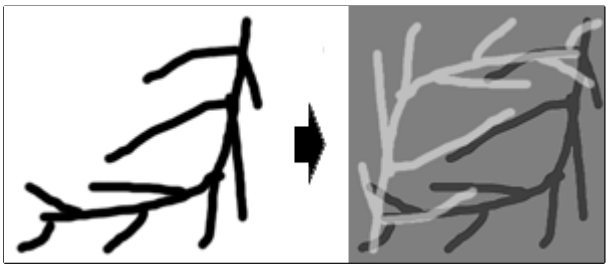


Рис. 4.1: Преобразование базового элемента в модели «произвольное вращение».

ние», приведены на с. 116 (рис. III.1, III.2).

Модель «сыр»

«Сыр» — пример простой текстуры, приближающей натуральный объект. Относительно реалистичная модель сыра — это множество «дырочек», хаотично разбросанных по равномерно окрашенной поверхности. Размер и ориентация «дырочек» произвольны. Модель можно упростить, ограничившись одной формой «дырочки» (это и будет базовый элемент), и лишь двумя слоями. Вероятность появления «дырочки» в точках репликации задается не очень высокой (чтобы «дырочек» в «сыре» оказалось не слишком много). Даже при сделанных упрощениях результат (рис. III.3, с. 117) выглядит вполне реалистично.

Модель «кирпичная стена»

«Идеальная» кирпичная стена является, как нетрудно заметить, регулярной структурой (т.е. все кирпичи одного размера, и кирпичи в соседних рядах смещены друг относительно друга на одно и то же значение). Однако большего реализма можно достигнуть, если в некоторых местах незначительно нарушить этот порядок.

Таблица 4.1: Модель «произвольное вращение»

<table>
<tr>
<td>
<table>
<tr><td>Поз.</td><td>1.0</td></tr>
<tr><td>Нег.</td><td>0.0</td></tr>
<tr><td>Нет</td><td>0.0</td></tr>
<tr><td>0°</td><td>0.25</td></tr>
<tr><td>90°</td><td>0.25</td></tr>
<tr><td>180°</td><td>0.25</td></tr>
<tr><td>270°</td><td>0.25</td></tr>
<tr><td>отраж.</td><td>0.0</td></tr>
<tr><td>слож.</td><td>1.0</td></tr>
<tr><td>ненул.н.</td><td>0.0</td></tr>
<tr><td>н.макс.</td><td>0.0</td></tr>
</table>
</td>
<td>0</td>
</tr>
<tr>
<td>0</td>
<td>
<table>
<tr><td>Поз.</td><td>1.0</td></tr>
<tr><td>Нег.</td><td>0.0</td></tr>
<tr><td>Нет</td><td>0.0</td></tr>
<tr><td>0°</td><td>0.25</td></tr>
<tr><td>90°</td><td>0.25</td></tr>
<tr><td>180°</td><td>0.25</td></tr>
<tr><td>270°</td><td>0.25</td></tr>
<tr><td>отраж.</td><td>0.0</td></tr>
<tr><td>слож.</td><td>1.0</td></tr>
<tr><td>ненул.н.</td><td>0.0</td></tr>
<tr><td>н.макс.</td><td>0.0</td></tr>
</table>
</td>
</tr>
</table>

В качестве базового элемента возьмем изображение «половинки» кирпича. В идеальном случае каждые два соседних элемента в одном ряду изображения должны быть повернуты друг относительно друга на 180°, а ряды сдвинуты друг относительно друга на один элемент или пол-элемента. В модели всего один слой. Добавим для каждого элемента небольшую вероятность быть ориентированным «неправильным» образом, например, как это сделано в таблице 4.2 (в таблице указаны только те события, которые имеют ненулевую вероятность). Результат показан на рис. III.4, слева (с. 117).

Реализм текстуры можно увеличить, если добавить в нее некоторый шум, создающий эффект неровности поверхности «кирпича». Для этого добавим в модель еще два слоя (номера слоев 2 и 3), на которых тот же самый базовый элемент будет реплицироваться с распределением, указанным в таблице 4.3 с небольшим весовым коэффициентом 0.2 (при весовом коэффициенте слоя 0 равным 1.0). Результат показан на рис. III.4, справа.

4.4 Управляющие маски слоев

Нетрудно заметить, что схема построения текстур, описанная выше, не приспособлена к локальной генерации, потому что использует генератор случайных чисел. Действительно, все изображения, сгенерированные по одной и той же модели и с одним и тем же базовым элементом, являются выборкой из одного и того же распределения, этой моделью определяемого, но не являются одинаковыми изображениями. В случае локальной генерации требуется независимо друг от друга получать различные фрагменты изображения. Нет никакой гарантии, что все такие фрагменты будут принад-

Таблица 4.2: Модель «кирпичная стена»

<table>
<tr>
<td><table><tr><td>Поз.</td><td>1.0</td></tr><tr><td>0°</td><td>0.8</td></tr><tr><td>90°</td><td>0.05</td></tr><tr><td>180°</td><td>0.15</td></tr><tr><td>ненул.н.</td><td>1.0</td></tr></table></td>
<td>0</td>
<td><table><tr><td>Поз.</td><td>1.0</td></tr><tr><td>0°</td><td>0.2</td></tr><tr><td>180°</td><td>0.8</td></tr><tr><td>ненул.н.</td><td>1.0</td></tr></table></td>
<td>0</td>
</tr>
<tr>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>0</td>
<td><table><tr><td>Поз.</td><td>1.0</td></tr><tr><td>0°</td><td>0.75</td></tr><tr><td>90°</td><td>0.05</td></tr><tr><td>180°</td><td>0.2</td></tr><tr><td>ненул.н.</td><td>1.0</td></tr></table></td>
<td>0</td>
<td><table><tr><td>Поз.</td><td>1.0</td></tr><tr><td>0°</td><td>0.1</td></tr><tr><td>90°</td><td>0.02</td></tr><tr><td>180°</td><td>0.88</td></tr><tr><td>ненул.н.</td><td>1.0</td></tr></table></td>
</tr>
<tr>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
</table>

Таблица 4.3: Модель «простой шум»

Поз.	0.33
Нег.	0.33
Нет	0.34
0°	0.25
90°	0.25
180°	0.25
270°	0.25
отраж.	0.5
слож.	1.0

лежать одному и тому же изображению. Более того, нет никакой гарантии, что две попытки получить один и тот же фрагмент дадут один и тот же результат. Таким образом, возникает задача модифицировать модель так, чтобы она допускала локальную генерацию.

Предлагается следующее решение: с помощью имеющихся таблиц событий вычислять события в каждой точке репликации *до* начала фазы непосредственной генерации. Может показаться, что такой подход сведет на нет все попытки создать действительно компактное представление непериодической текстуры. Появляются структуры данных конечного размера, в которых указаны уже не вероятности, а коды определенных событий. В этом случае приходится либо мириться с неизбежным периодическим эффектом, либо генерировать структуры большого размера, что соответствует большим периодам текстуры (а это приведет к увеличению объема памяти для хранения представления). Однако, как будет показано ниже, нетрудно найти такое соотношение между размером периода и объемом памяти, которое даст удовлетворительный результат.

Поскольку в предложенной модели слои генерируются независимо друг от друга, структура, хранящая информацию об уже рассчитанных событиях — это набор двумерных массивов. Каждый массив соответствует одному слою. Каждый элемент массива соответствует одной точке репликации. Такой массив будем называть *управляющей маской слоя* (УМС).

Заметим, что для хранения информации о любом событии в точке репликации достаточно одно байта. Действительно, 2 бита кодируют одно из событий «позитивная репликацию», «негативная репликация» и «нет репликации», 2 бита кодируют один из четырех углов поворота, 1 бит ко-

дирует отражение (или транспонирование), 2 бита кодируют операцию добавления. Итого 7 битов.

С помощью УМС возможна как генерация изображения целиком, так и локальная генерация. В первом случае процесс формирования изображения мало чем отличается от описанного ранее, за тем лишь исключением, что каждое событие не вычисляется на этапе формирования, вместо этого используется код предварительно рассчитанного события. Во втором случае выбираются коды только тех событий, которые соответствуют точкам репликации, расположенным в окрестности того фрагмента, который требуется восстановить. Поскольку размещение УМС в пространстве фиксируется, а построение по УМС уже не содержит элемента случайности, то все возможные независимо полученные фрагменты гарантированно принадлежат одному и тому же изображению.

Заметим, что введение УМС не только решило проблему локальной генерации, но и разбило сам процесс на две фазы: расчет УМС и отображение текстуры на основе готовых УМС. Заметим, что только вторую фазу необходимо реализовать как можно более эффективно, в реальном времени и, возможно, аппаратно. В то же время расчет УМС содержит основной объем вычислений. На самом деле эти вычисления относительно просты, но возможны различные усложнения и усовершенствования модели, требующие более сложных расчетов. Однако, если результат этих расчетов можно вывести в виде УМС, то отображение соответствующего объекта окажется ничуть не менее эффективным. В любом случае отображение объекта с помощью УМС — это простой неинтеллектуальный алгоритм, использующий тривиальные структуры входных данных — двумерные массивы байтов.

Недостатком УМС является их конечный размер. Действительно, при использовании таблиц событий (без УМС) возможно генерировать текстуры любого размера, причем нет необходимости заранее определять этот размер. Кроме того, объем памяти, необходимый для хранения таблиц событий, никак не зависит от размера генерируемого изображения. При использовании УМС также можно генерировать текстуру любого размера. Однако, в силу конечности размера УМС, всегда можно потребовать построить текстуру такого размера, который не сможет быть полностью покрыт УМС, и это приведет к периодизации. Очевидно, что чем больше размер УМС, тем больше период, однако, увеличение размера УМС увеличивает размер представления текстуры.

Заметим, однако, что на практике даже относительно небольшие УМС обеспечивают достаточно большой период. Действительно, если, например, модель состоит только из одного слоя 0, которому соответствует базовый элемент размера 32×32 пиксела, то УМС размера 64×64 гарантирует создание изображения без периодизации размером до 1024×1024 пикселов. (Напомним, что расстояние между двумя точками репликации равно половине размера стороны базового элемента, поэтому получаем $64 \cdot (32/2) = 1024$.

Может показаться, что чем меньше размер базового элемента, тем больше должен быть размер УМС. Например, если размер базового элемента 16×16 пикселов, то для генерации изображения без периодизации размером 1024×1024 пикселов необходима УМС размера 128×128. Однако если модель содержит более одного слоя, то вовсе не обязательно, чтобы все УМС полностью покрывали изображение. Например, модель состоит из двух слоев (слой 0 и слой 1), размер базового элемента для слоя 0 — 32×32 пиксела.

Тогда базовый элемент для слоя 1 будет иметь размер 16×16 пикселов. Теперь предположим, что УМС обоих слоев имеют размер 64×64 и требуется сгенерировать текстуру размера 1024×1024. Для слоя 1 будет иметь место периодизация (для покрытия изображения потребуется 4 копии УМС слоя 1), но слой 1 будет объединен со слоем 0, который не имеет периодизации, и в выходном изображении периодическая структура слоя 1 будет, вероятнее всего, мало различима. Еще лучшие результаты могут быть получены, если размеры УМС разных уровней не являются степенями двойки, не имеют общих делителей и не являются квадратными. Предположим, например что УМС слоев 0 и 1 имеют размеры соответственно 50×60 и 83×71. Очевидно, что период результирующего изображения будет существенно больше, чем период каждого из слоев в отдельности.

Возможны и другие способы увеличения периода.

4.5 Представление данных и особенности реализации

Как отмечалось выше, использование УМС разделяет процесс генерации на две фазы.

Тонкости реализации первой фазы — генерации УМС — здесь не будут обсуждаться. Можно только заметить, что, поскольку таблицы событий задаются для каждого слоя независимо, расчет нескольких УМС элементарно распараллеливается.

Рассмотрим представление текстуры после генерации УМС. Разумеется, такое представление должно быть возможно более компактным и легко интерпретируемым.

Базовый элемент представлен 8-разрядной битовой матрицей определен-

ного размера (в наших экспериментах обычно 128×128, 64×64 или 32×32 пиксела). Однако кроме самого́ базового элемента требуется иметь его копии низкого разрешения. В наших экспериментах использовались два подхода. Первый — представление изображения в виде преобразования Хаара, которое требует ровно столько же места, сколько и исходное изображение, но позволяет достаточно быстро восстановить изображения с требуемым уровнем разрешения. Второй подход — хранение всех требуемых копий в явном виде. Этот способ приводит к большему расходу памяти, но обеспечивает более быстрое построение.

В качестве примера обратимся к модели «кирпичная стена». Модель состоит из трех слоев (0, 2 и 3). Размер УМС слоя 0 — 40×40 байтов (структура УМС была рассмотрена в п. 4.4). Для оставшихся слоев используется одна и та же УМС размера 20×20 байтов. Размер базового элемента 32×32 пиксела. Кроме того, если рассматривается случай явного представления копий базового элемента, то дополнительно требуется хранить копии размера 8×8 и 4×4 пикселов. Каждый пиксел кодируется одним байтом. Таким образом, все описание текстуры занимает $40^2 + 20^2 + 32^2 + 8^2 + 4^2 = 3104$ байтов плюс не более 20 байтов для дополнительной информации (весовых коэффициентов, интенсивности фона и пр.). Этой информации достаточно, чтобы сгенерировать текстуру с периодом 640×640 пикселов. Изображение такого же размера, представленного в виде 8-битной матрицы занимает 409600 байтов, т.е. приблизительно в 130 раз больше. Для возможности масштабирования текстуры требуется добавить недостающие копии базового элемента размера 16×16 и 2×2 пиксела. Кроме того, УМС для слоев 2 и 3 могут быть разными. Но даже в этом случае размер представления не

превысит 3800 байтов, что в 107 раз меньше, чем представление в явном виде (т.е. в виде матрицы).

Представление может быть еще более компактным. В ряде случаев структура УМС позволяет применить к ним простые методы сжатия, которые, практически не влияя на эффективность генерации текстуры, могут существенно уменьшить объем представления. В большинстве УМС, использованных в наших опытах, расположение нулевых элементов (т.е. элементов, соответствующих событию «нет репликации») имеют регулярную структуру (например, в модели «произвольное вращение» они расположены в шахматном порядке). Если эта структура известна, то нулевые элементы можно не кодировать.

Теперь обратим внимание на процесс генерации и обсудим процесс вычисления произвольного пиксела текстуры.

Поскольку основной объем данных представлен в виде двумерных байтовых массивов, то доступ к любому элементу любой УМС и к любому пикселу базового элемента любого разрешения тривиален. (Более сложным образом осуществляется доступ к пикселам базового элемента в том случае, если элемент хранится в виде преобразование Хаара). Если заданы координаты пиксела, интенсивность которого следует рассчитать, то не составляет труда найти в каждой УМС модели все элементы, соответствующие точкам репликации, влияющим на формирование данного пиксела. Далее следует найти пикселы всех таких репликаций, соответствующие заданной координате генерируемого изображения. Эти пикселы определяются согласно координатам точек репликации и кодам событий в этих точках. Заметим, что почти для всех событий вычисление пиксела реплика-

ции сводится к поиску соответствующего пиксела копии базового элемента требуемого разрешения (исключение составляет событие «негативная репликация», оно требует взять интенсивность найденного пиксела с противоположным знаком).

Для простоты изложения предположим, что при добавлении репликаций используется только операция простого сложения. Согласно определению модели, в любой точке генерируемого изображения могут пересечься не более четырех репликаций одного слоя. Следовательно, на каждом слое требуется выполнить не более трех сложений. Результат умножается на весовой коэффициент (1 умножение). Для группировки результатов по N слоям требуется $N - 1$ сложение. К полученному нужно прибавить интенсивность фона (1 сложение). Таким образом, для вычисления одного пиксела N-слойной текстуры требуется не более $4N$ сложений и 1 умножение. Число операций, естественно, увеличивается, если масштабированные копии базового элемента хранятся не в явном виде, и требуется декодирование.

Так же, как и в случае генерации УМС, значительная часть вычислений выполняется для каждого слоя независимо, что дает возможность распараллелить процесс.

4.6 Связь с теорией вейвлет-анализа

В основу предложенной структуры репликации положено разложение сигнала по вейвлет-базису (1.2), точнее, его расширение на двумерный случай (A.17), описанное в Приложении А, п. А.4 (с. 126).

Запишем следующую формулу:

$$f(x,y) = f^{(0)}(x,y) + \sum_{i=0}^{N-1} \sum_{j,k=-\infty}^{+\infty} w_{j,k}^{(i)}\, \psi(2^i x - j, 2^i y - k). \qquad (4.4)$$

От формулы (A.17) она отличается следующим. Во-первых, суммирование происходит по конечному числу уровней разрешения. Во-вторых, вместо трех порождающих вейвлетов есть только один. Последнее, очевидно, не дает возможности говорить об обратимости преобразований. Однако в данном случае эти вопросы не являются актуальными. В данной задаче формируется вейвлет-образ (или некое подобие вейвлет-образа) объекта, потом выполняется обратное преобразование, и таким образом объект создается. Поскольку прообраза объекта никогда не существовало, вряд ли возможно говорить об обратимости.

В качестве порождающего вейвлета $\psi(\bullet, \bullet)$ выступает базовый элемент. Начальным приближением $f^{(0)}(\bullet, \bullet)$ является константная функция v — интенсивность фона. Аналогами вейвлет-коэффициентов в модели выступают два объекта. Во-первых, это весовые коэффициенты $w^{(i)}$, определяемые по одному для каждого уровня разрешения (уровню разрешения соответствует слой модели). Во-вторых, коэффициент при каждой копии базового элемента меняется на функционал $\mathrm{W}_\xi[\bullet]$, который преобразует аргумент в зависимости от значения случайной величины ξ, распределение которой задается параметрами конкретной модели. Преобразования базового элемента соответствуют допустимым для моделям событиям. Для функционала возможна и иная запись: $\mathrm{W}[\bullet, c]$, где c — числовой параметр. То есть, вместо случайной величины действие функционала на аргумент определяется числом, полученным заранее некоторым образом[2]. С точки зрения модели, такая

[2]Очевидно, что обычное вейвлет-преобразование является частным случаем такого обобщения с функ-

ситуация соответствует применению УМС, и в этом случае параметрами функционалов являются коды событий, т.е. элементы УМС, и их в определенной степени можно считать обобщениями вейвлет-коэффициентов.

Таким образом, вместо (4.4) получаем:

$$f(x,y) = v + \bigoplus_{i=0}^{N-1} w^{(i)} \left(\bigoplus_{j,k=-\infty}^{+\infty} \mathrm{W}\left[\psi(2^i x - j, 2^i y - k),\, c_{j,k}^{(i)}\right] \right). \qquad (4.5)$$

Символы $\oplus$ использованы вместо Σ, чтобы показать, что могут быть использованы операции похожие, но не идентичные обычному сложению. Кроме того, нужно отметить, что формула не учитывает, что эти операции могут быть не коммутативными, а выбор операции также, вообще говоря, зависит от параметра $c_{j,k}^{(i)}$.

4.7 Анализ результатов. Развитие задачи

В данной главе рассмотрена модель для представления стохастических текстур, ориентированная на использование в графических устройствах на аппаратном уровне. Модель основана на обобщении разложения сигнала по вейвлет-базису. Указанное представление компактно, масштабируемо, допускает локальное декодирование. Разработан быстрый алгоритм декодирования такого представления и нанесения текстуры, допускающий аппаратную реализацию.

В заключение укажем некоторые пути развития задачи. В первую очередь, это различные модификации существующей модели, расширяющие класс текстур, которые с помощью модели могут быть сгенерированы. Это, например, ослабление ограничений, накладываемых на размер базового

ционалом $\mathrm{W}[u(\bullet), c] = c\,u(\bullet)$, где $u(\bullet)$ — вейвлет, а c — вейвлет-коэффициент в обычном понимании.

элемента, на расположение точек репликации и т.д. Следует, однако, заметить, что подобные усовершенствования могут привести к увеличению объема данных, а также ухудшить масштабируемость текстуры.

К более серьезным расширениям можно отнести изменение вероятностной модели, например, на основе полей Маркова. Как уже было отмечено, предложенная двухэтапная схема (с использованием УМС) позволяет изменять алгоритм вычисления УМС, не меняя при этом этап генерации.

Также возможно ввести зависимости между слоями (в предложенной модели все слои обрабатываются независимо друг от друга). Однако это может потребовать усложнения структуры данных и увеличения их объема.

Предложенный выше метод включает в себя только фазу *синтеза* изображения — по заданным параметрам модели генерируется текстура, при этом сами параметры подбираются пользователем произвольно. В ряде случаев удается «угадать» параметры так, чтобы получилось относительно реалистическое изображение. В связи с этим представляет интерес новое направление в развитии задачи — разработка фазы *анализа*. Для заданного образца некоторой текстуры предлагается отыскать параметры модели (возможно, конечно, что сама модель будет несколько изменена): выделить один или несколько базовых элементов и подобрать параметры их распределения так, чтобы сгенерирована на основе полученных данных текстура была бы визуально близка к оригиналу.

Если фаза анализа будет разработана, то полученный в результате метод может быть использован не только для создания новых текстур, но и для решения задачи, аналогичной тем, что решались в работах, перечисленных в начале главы: по заданному образцу построить текстуру произ-

вольного размера. Можно ожидать, что по качеству генерируемых объектов такой метод будет уступать вышеупомянутым подходам, однако он будет обладать одним несомненным преимуществом: фаза синтеза (генерации) будет по-прежнему отделена от достаточно нетривиальной фазы анализа, она сможет выполняться в реальном времени и даже реализована аппаратно.

Заключение

В работе рассмотрено несколько задач компьютерной графики: построение адаптивных сеточных представлений для объектов, параметризуемых на плоскости (в т.ч. функций освещенности поверхности); построение линий уровня таблично заданной функции двух переменных; генерация стохастических текстур. В основе решения этих задач лежит единый подход — многомасштабное представление информации.

Показано, что многомасштабные представления предоставляют широкие возможности для анализа и синтеза графических объектов различной природы (растровых изображений, кривых, поверхностей).

Основным инструментом для работы с многомасштабными представлениями является вейвлет-анализ. Алгоритмы вейвлет-преобразований дискретных сигналов просты и эффективны в реализации. Однако вейвлет-преобразования обладают рядом ограничений, которые, как оказалось, не всегда позволяют получить необходимый результат. Так, для решения задачи реконструкции освещенности потребовалось изменить алгоритм преобразования. Это привело к дополнительному (хотя и небольшому) расходу памяти при вычислениях, однако ощутимо улучшило сглаживающие свойства преобразования, что для данной задачи являлось наиболее существенным требованием. В задаче генерации текстур упомянутые алгорит-

мы вейвлет-преобразований вообще не используются, однако показано, что структура предложенной модели имеет структуру, близкую к разложению двумерного сигнала по базису вейвлетов.

В процессе работы над диссертацией предложена систематизация различных видов вейвлет-преобразований, применяемых в прикладных задачах, проведено сравнение имеющихся в различных источниках форм записи преобразований и алгоритмов, реализованных на их основе, результаты опубликованы в [13].

Основные результаты диссертации состоят в следующем:

1. Разработан и реализован метод построения адаптивных треугольных сеточных представлений объектов, параметризуемых на плоскости. Метод основан на древовидной структуре многомасштабного анализа информации. С помощью предложенного метода решена задача расчета и представления функций освещенности.

2. Разработан и реализован метод построения линий уровня (изолиний) на основе B-сплайнового вейвлет-преобразования. Метод обеспечивает сглаживание кривых, построенных по зашумленным данным, и адаптацию к параметрам области отображения.

3. Предложена модель для представления масштабируемых стохастических текстур, основанная на обобщении разложения сигнала по вейвлет-базису. Разработан быстрый алгоритм декодирования такого представления и нанесения текстуры, допускающий аппаратную реализацию.

4. Применено на практике свойство многофункциональности многомас-

штабных представлений: одно и то же представление объекта используется для решения целого ряда подзадач (сглаживание, масштабирование, сжатие информации и пр.), что упрощает общую процедуру обработки и повышает эффективность ее реализации.

5. На примере разработанных методов показано, что разнообразие форм вейвлет-преобразований позволяет, в зависимости от структуры объекта и требований к приложению, делать выбор между простыми, производительными алгоритмами и более сложными методами, которые обеспечивают лучшие результаты обработки, но менее эффективны в реализации.

Иллюстрации

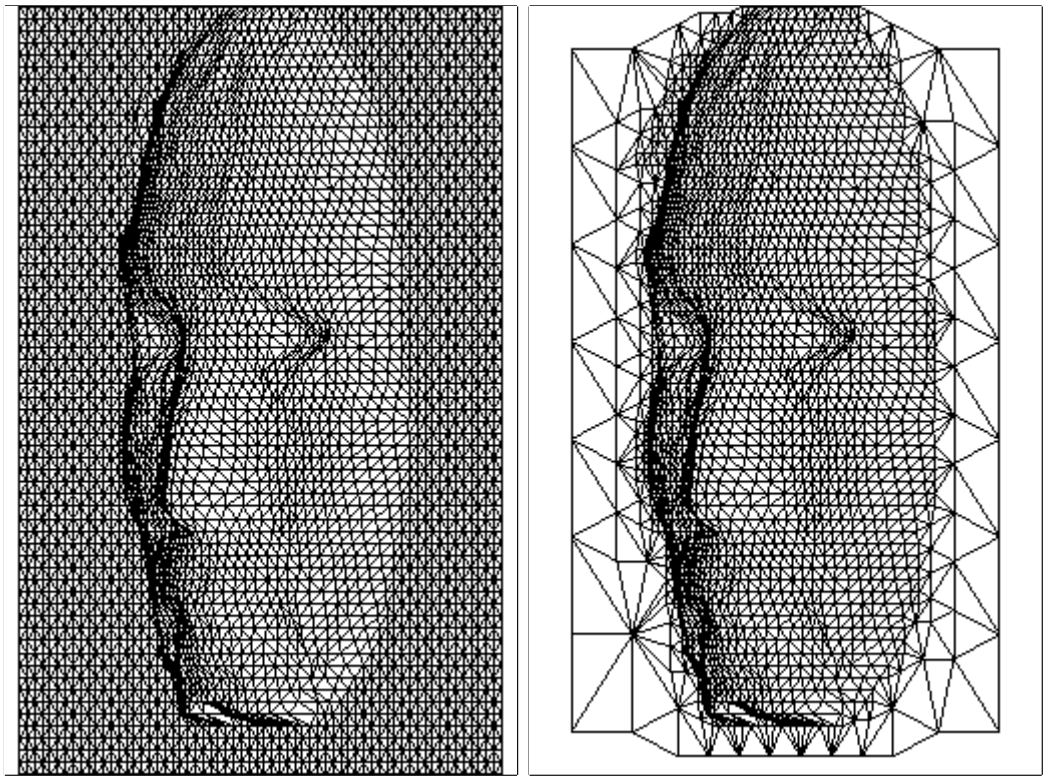


Рис. I.1: Слева исходная сетка 4096 (64×64) уз., 7938 тр.; справа сетка 2046 уз., 4120 тр.

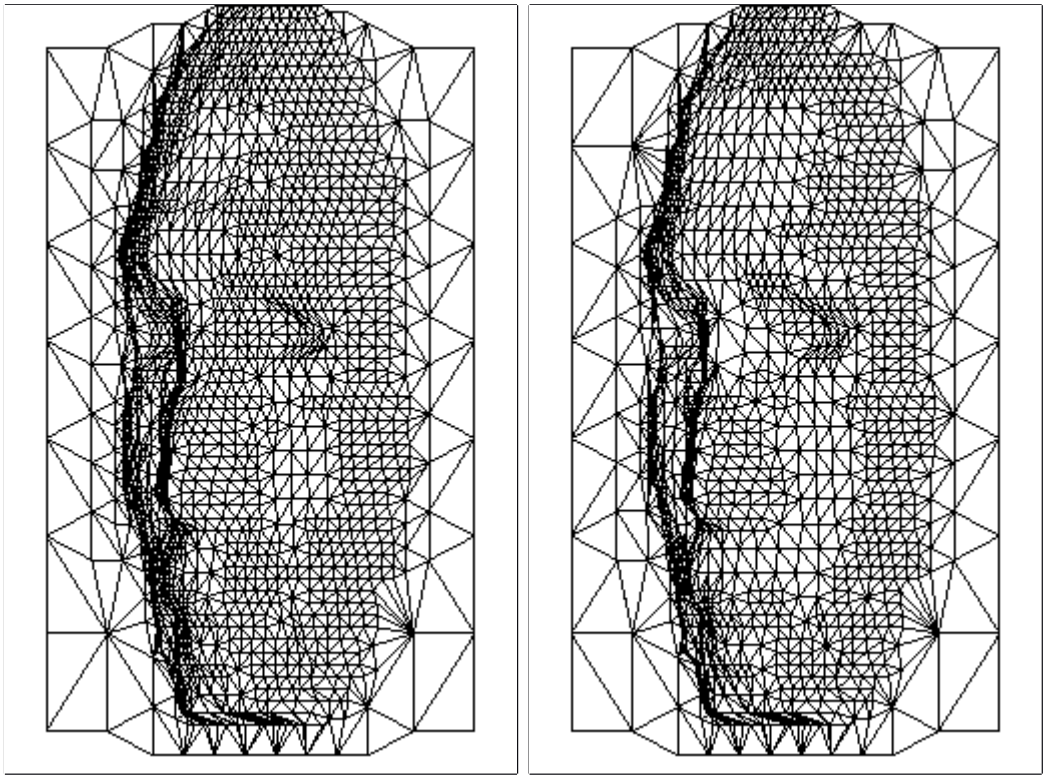


Рис. I.2: Слева сетка 1651 уз., 3252 тр.; справа сетка 1396 уз., 2744 тр.

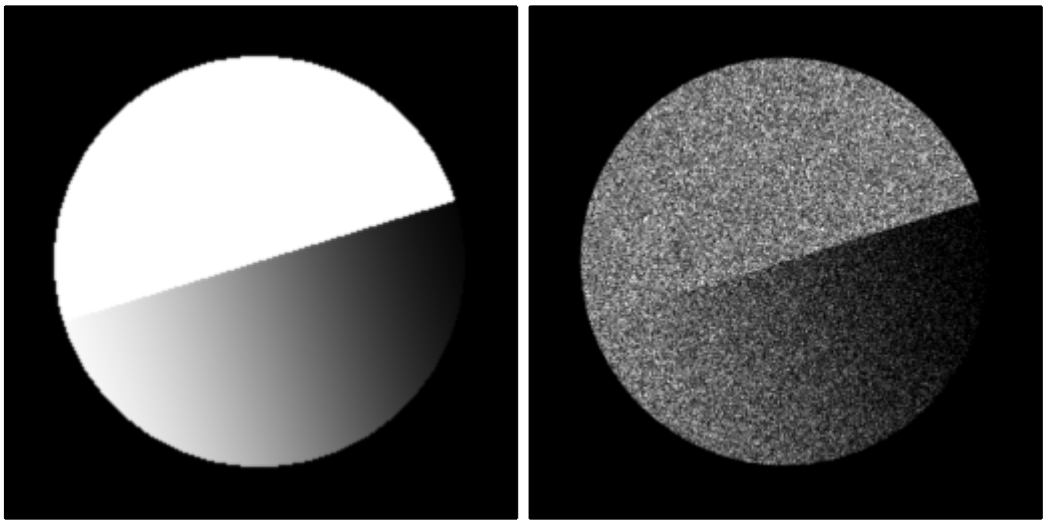


Рис. I.3: Эталонное изображение (слева) и рез-т попадания 200 тыс. фотонов (справа).

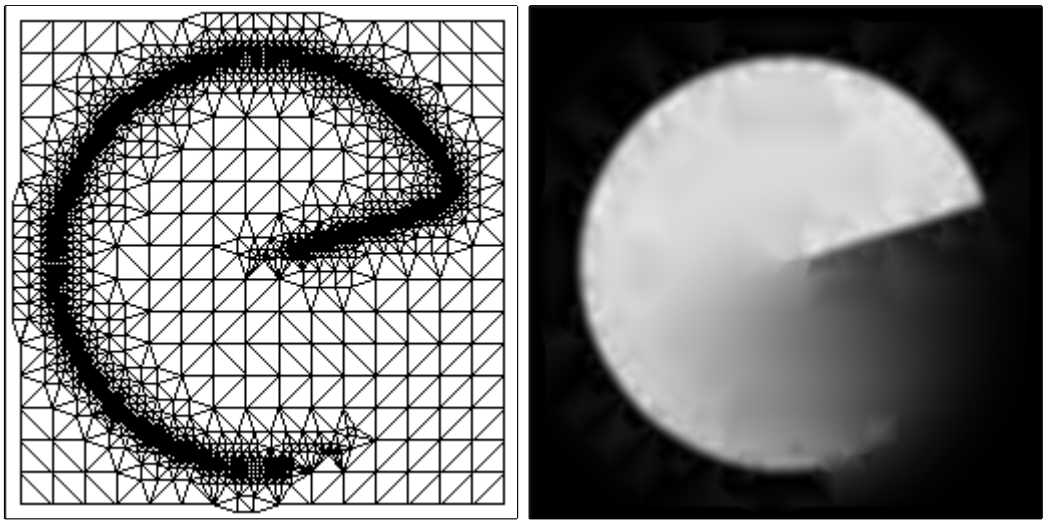


Рис. I.4: Сетка (слева) и итоговое изображение (справа).

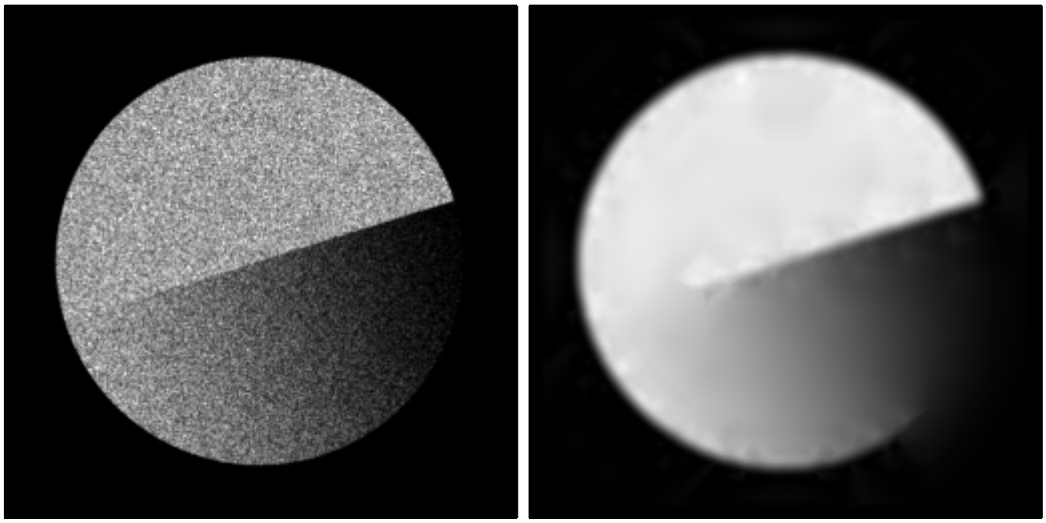


Рис. I.5: Реконструкция по 500 тыс. фотонам.

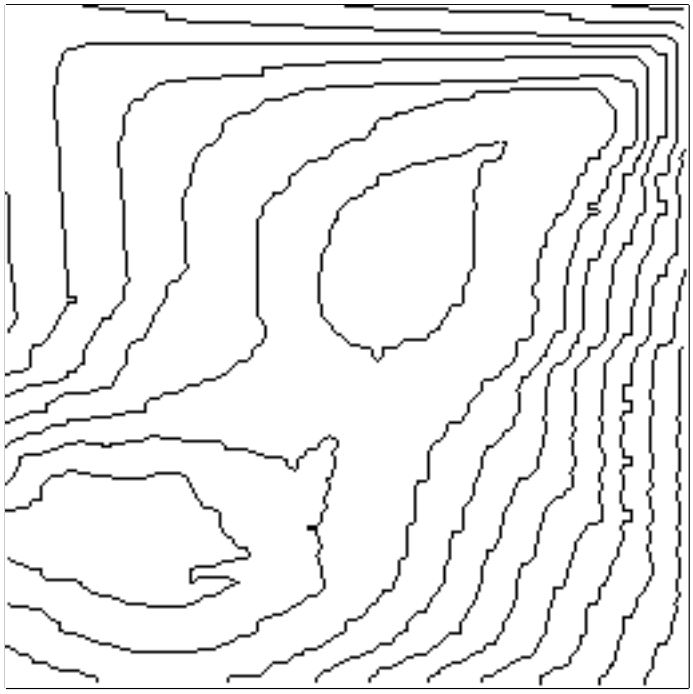


Рис. II.1: Построение линий по исходным данным без сглаживания

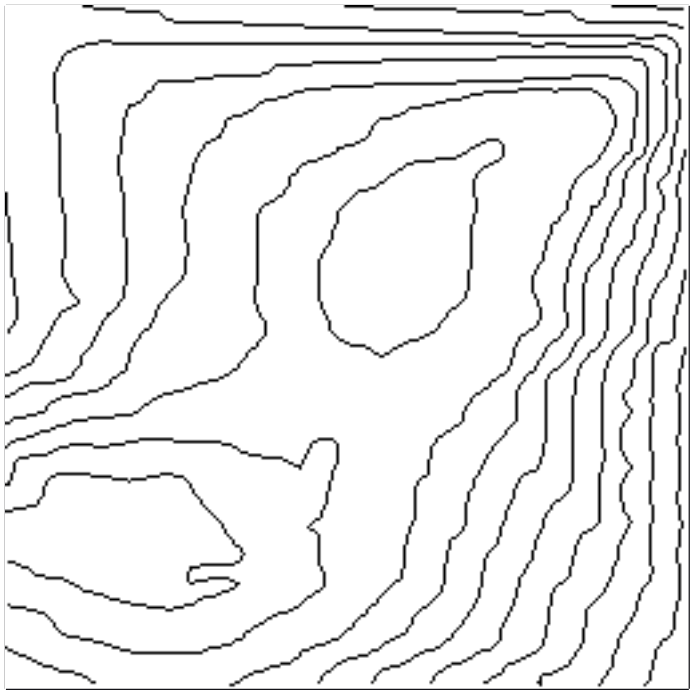


Рис. II.2: Сглаживание (2 шага, основной фильтр)

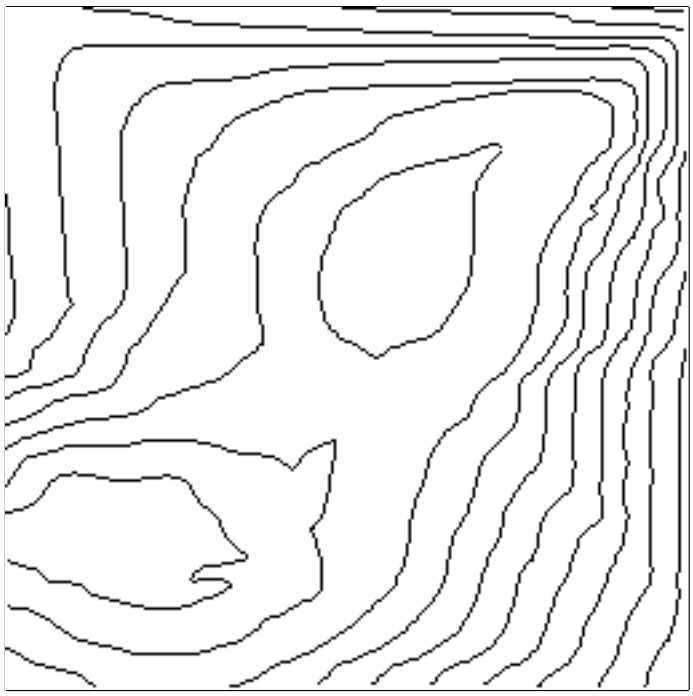


Рис. II.3: Сглаживание (2 шага, дополнительный фильтр)

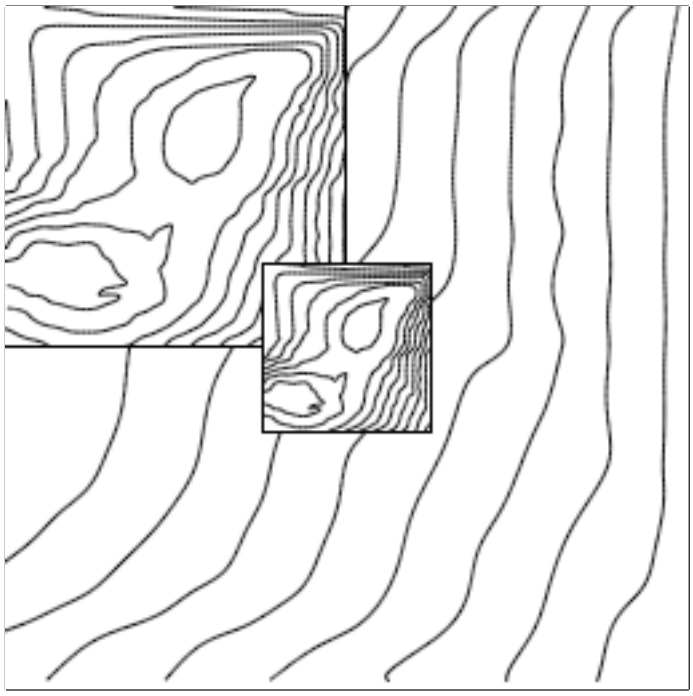


Рис. II.4: Подавление лестничного эффекта и масштабирование

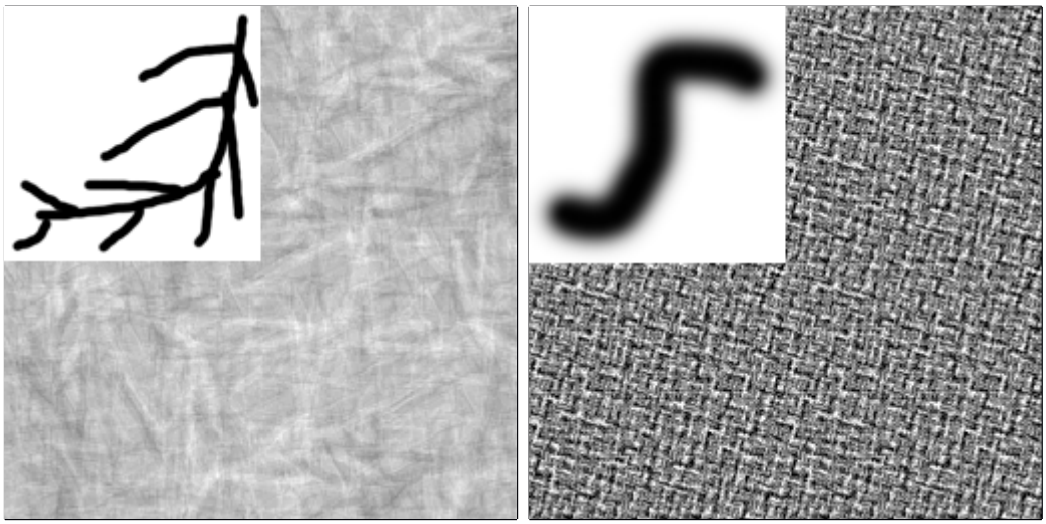


Рис. III.1: Модель «произвольное вращение» с разными базовыми элементами

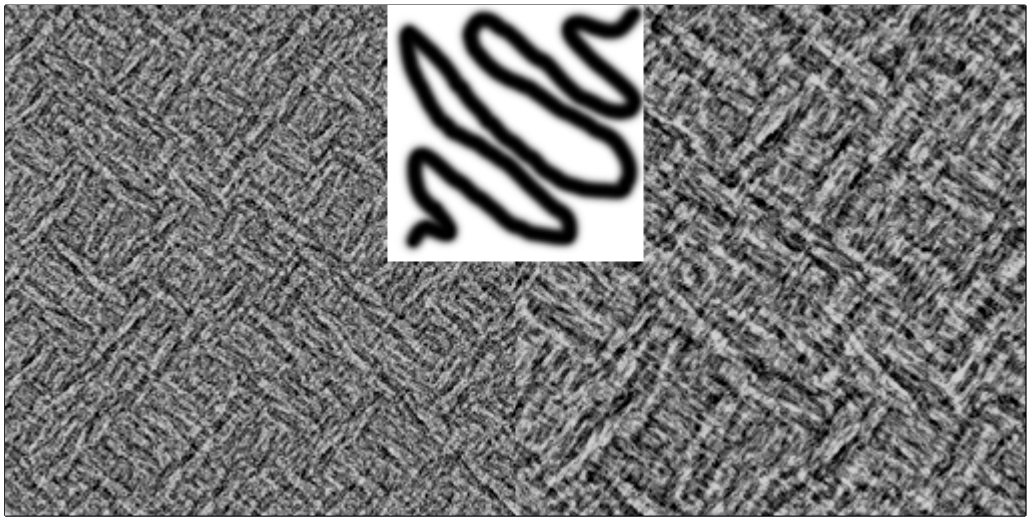


Рис. III.2: Модель «произвольное вращение». Текстура показана в двух масштабах

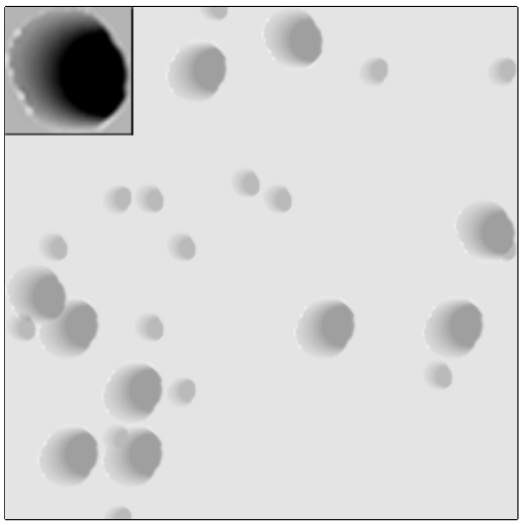


Рис. III.3: Модель «сыр» с базовым элементом

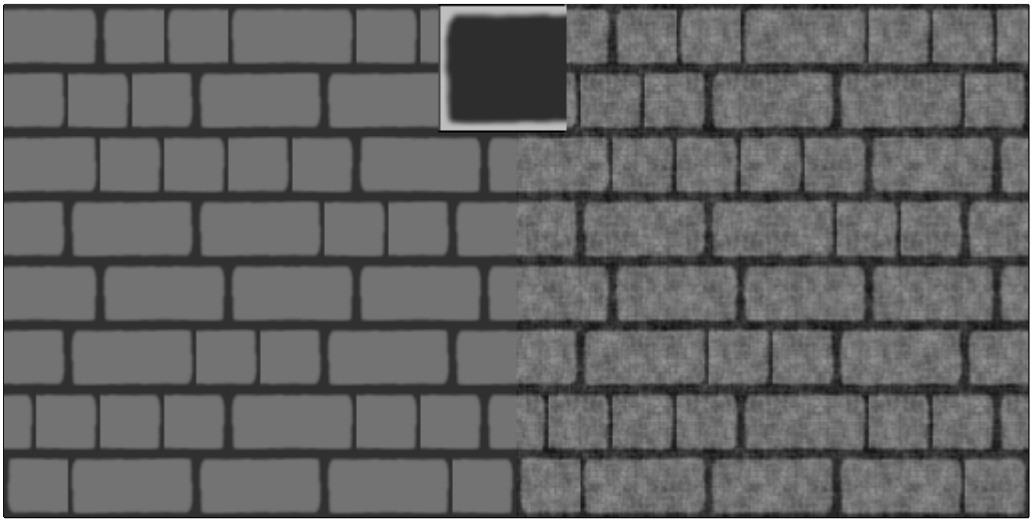


Рис. III.4: Модель «кирпичная стена» с базовым элементом (слева первоначальный вариант, справа вариант с добавлением шума)

Приложение А

Многомасштабный анализ и вейвлет-преобразования

А.1 Ортогональный многомасштабный анализ

Ортогональным многомасштабным анализом в $\mathbf{L}_2(\mathbf{R})$ называется последовательность замкнутых подпространств $V^{(i)} \subset \mathbf{L}_2(\mathbf{R}),\ i \in \mathbf{Z}$, таких что:

1. $V^{(i)} \subset V^{(i+1)}, \quad i \in \mathbf{Z}$.
2. $\bigcup\limits_{i\in\mathbf{Z}} V^{(i)}$ плотно в $\mathbf{L}_2(\mathbf{R})$.
3. $\bigcap\limits_{i\in\mathbf{Z}} V^{(i)} = \emptyset$.
4. $v(x) \in V^{(i)} \iff v(2x) \in V^{(i+1)},\ i \in \mathbf{Z}$.
5. $v(x) \in V^{(0)} \iff v(x-j) \in V^{(0)},\ j \in \mathbf{Z}$.
6. $\exists\, \varphi(x) \in V^{(0)}$, $\int_{-\infty}^{\infty} \varphi(x)\,dx \neq 0$: последовательность $\{\varphi(x-j)\}_{j\in\mathbf{Z}}$ является ортонормированным базисом (ОНБ) в $V^{(0)}$. Элемент $\varphi(x)$ называется *порождающей скейлинг-функцией.*

Непосредственно из определения вытекают следующие свойства:

1. $\exists\, h_k \in \mathbf{R},\ k \in K,\ K \subset \mathbf{Z}$:

$$\varphi(x) = \sqrt{2} \sum_{k \in K} h_k\, \varphi(2x - k). \tag{A.1}$$

Это выражение называется *масштабным соотношением* (или *масштабным уравнением*) для скейлинг-функций.

2. $\forall\ i \in \mathbf{Z}$ последовательность $\left\{\varphi_j^{(i)}(x) \equiv \sqrt{2^i}\, \varphi(2^i x - j)\right\}_{j \in \mathbf{Z}}$ является ОНБ в пространстве $V^{(i)}$. Функции $\varphi_j^{(i)}(x)$ называются *скейлинг-функциями.*

3. Если $\varphi(x) \in \mathbf{L}_1(\mathbf{R})$, и $\int_{-\infty}^{+\infty} \varphi(x)\, dx = 1$, то с точностью до значений на множестве меры нуль эта функция единственным образом определяется масштабным соотношением (A.1), т.е. набором значений $\{h_k\}_{k \in K}$[1].

Для каждой пары подпространств $V^{(i)} \subset V^{(i+1)},\ i \in \mathbf{Z}$, многомасштабного анализа должно существовать подпространство $W^{(i)}$, такое что

$$V^{(i)} \perp W^{(i)},$$

$$V^{(i+1)} = V^{(i)} \oplus W^{(i)}.$$

Такие подпространства называют *уточняющими* или *детализирующими* в том смысле, что они содержат уточняющую информацию, необходимую для перехода от уровня разрешения i к уровню $i + 1$. Справедливо следующее:

$$\bigoplus_{i=-\infty}^{+\infty} W^{(i)} = \mathbf{L}_2(\mathbf{R}).$$

Если существует элемент $\psi(x) \in W^{(0)}$ такой, что последовательность $\{\psi(x - j)\}_{j \in \mathbf{Z}}$ является ОНБ в $W^{(0)}$, то этот элемент называется *порождающим вейвлетом.*

[1] как известно, значения на множестве меры нуль не влияют на результат интегрирования, поэтому такая точность определения функций является достаточной.

Если $\psi(x) \in W^{(0)}$ — порождающий вейвлет, то набор функций $\left\{\psi_j^{(i)}(x) \equiv \sqrt{2^i}\psi(2^i x - j)\right\}_{i,\,j\in\mathbf{Z}}$ образует ОНБ в $\mathbf{L}_2(\mathbf{R})$. Функции из этого набора называются *вейвлетами*. Детализирующие подпространства $W^{(i)}$, $i \in \mathbf{Z}$, принято также называть *вейвлет-пространствами*.

Очевидно, что порождающий вейвлет $\psi(x)$ является элементом пространства $V^{(1)}$. Следовательно, найдутся такие числа $g_l \in \mathbf{R},\ l \in L,\ L \subset \mathbf{Z}$, что

$$\psi(x) = \sqrt{2}\sum_{l\in L} g_l\,\varphi(2x - k). \tag{A.2}$$

Это соотношение является *масштабным соотношением* для вейвлетов. В отличие от соотношения (A.1), оно не является уравнением.

Таким образом, порождающий вейвлет $\psi(x)$ с точностью до значений на множестве меры нуль определяется коэффициентами $\{g_l\}_{l\in L}$, если определена порождающая скейлинг-функция $\varphi(x)$, а она, в свою очередь, определяется коэффициентами $\{h_k\}_{k\in K}$ соотношения (A.1). Следовательно, система скейлинг-функций и вейвлетов может быть полностью определена двумя наборами коэффициентов $\{h_k\}_{k\in K}$ и $\{g_l\}_{l\in L}$.

Замечание. Всегда можно считать, что $K = \mathbf{Z}$ и $L = \mathbf{Z}$, т.е. коэффициенты в наборах определены для любого целого индекса. Если это не так, то наборы доопределяются на всем множестве целых индексов нулевыми элементами.

Так как набор функций $\left\{\psi_j^{(i)}(x)\right\}_{i,j\in\mathbf{Z}}$ является ОНБ в $\mathbf{L}_2(\mathbf{R})$, то любую функцию $f(x) \in \mathbf{L}_2(\mathbf{R})$ можно единственным образом представить в виде разложения

$$f(x) = \sum_{i=-\infty}^{+\infty}\sum_{j=-\infty}^{+\infty} w_j^{(i)}\,\psi_j^{(i)}(x), \tag{A.3}$$

где

$$w_j^{(i)} = \left\langle f(\bullet) \;\middle|\; \psi_j^{(i)}(\bullet) \right\rangle, \quad \psi_j^{(i)}(x) = \sqrt{2^i}\psi(2^i x - j), \quad i, j \in \mathbf{Z}. \tag{A.4}$$

Набор вейвлет-коэффициентов, полученных по формулам (A.4) называется *диадным* или *дискретным ортогональным вейвлет-преобразованием* сигнала $f(x)$. Формула (A.3) определяет *обратное* диадное ортогональное вейвлет-преобразование (ср. с (1.1), с. 17).

Значения $w_j^{(i)},\ i, j \in \mathbf{Z}$, называются *детализирующими коэффициентами* или *вейвлет-коэффициентами.*

Заметим, что для любого $i_0 \in \mathbf{Z}$

$$\bigoplus_{i=-\infty}^{i_0-1} W^{(i)} = V^{(i_0)},$$

следовательно,

$$\mathbf{L}_2(\mathbf{R}) = \bigoplus_{i=-\infty}^{+\infty} W^{(i)} = V^{(i_0)} \oplus \bigoplus_{i=i_0}^{+\infty} W^{(i)}.$$

В пространстве $V^{(i_0)}$ существует базис скейлинг-функций, следовательно, набор функций

$$\left\{\varphi_j^{(i_0)}(x), \quad \psi_j^{(i)}(x)\right\}_{i,j\in\mathbf{Z},\ i\geq i_0}$$

также является ОНБ в $\mathbf{L}_2(\mathbf{R})$ (будем называть такой базис *комбинированным*). Тогда справедливо следующее представление (ср. с (1.2), с. 17):

$$f(x) = \sum_{j=-\infty}^{+\infty} v_j^{(i_0)}\varphi_j^{(i_0)}(x) + \sum_{i=i_0}^{+\infty}\sum_{j=-\infty}^{+\infty} w_j^{(i)}\psi_j^{(i)}(x), \quad i_0 \in \mathbf{Z}, \tag{A.5}$$

где

$$v_j^{(i_0)} = \left\langle f(\bullet) \;\middle|\; \varphi_j^{(i_0)}(\bullet) \right\rangle, \quad \varphi_j^{(i_0)}(x) = \sqrt{2^{i_0}}\varphi(2^{i_0}x - j), \quad i_0, j \in \mathbf{Z}.$$

Представление (A.5) можно рассматривать, как разложение сигнала $f(x)$ на две проекции — проекцию на пространство $V^{(i_0)}$ (первое слагаемое формулы), и проекцию на ортогональное дополнение $V^{(i_0)}$ до $\mathbf{L}_2(\mathbf{R})$ (второе слагаемое). Структура пространств такова, что проекция сигнала на первое

пространство является огрубленным (или, пользуясь терминологией анализа Фурье, *низкочастотным*) представлением этого сигнала, а на второе — высокочастотным, т.е. содержащим уточняющую (детализирующую) информацию о сигнале, потерянную при проецировании на пространство $V^{(i_0)}$. Очевидно, что чем выше значение i_0, тем больше информации, содержащейся во втором слагаемом формулы (A.5), «перетекает» в первое.

Проекцию сигнала на пространство $V^{(i_0)}$ будем называть представлением (или приближением) сигнала с разрешением i_0. Кроме собственно проекции, так можно называть и набор коэффициентов $\{v_j^{(i_0)}\}_{j\in\mathbf{Z}}$ разложения этой проекции по базисным скейлинг-функциям, т.к. при фиксированном базисе скейлинг-функций коэффициенты однозначно определяют такое приближение.

Далее будем для краткости пользоваться следующими обозначениями для последовательностей подпространств и функций:

$$\mathbf{V} \equiv \left\{V^{(i)}\right\}_{i\in\mathbf{Z}}; \qquad \mathbf{W} \equiv \left\{W^{(i)}\right\}_{i\in\mathbf{Z}};$$

$$\mathbf{\Phi} \equiv \left\{\varphi_j^{(i)}(x)\right\}_{i,j\in\mathbf{Z}}; \quad \mathbf{\Phi}^{(i)} \equiv \left\{\varphi_j^{(i)}(x)\right\}_{j\in\mathbf{Z}}, \quad i \in \mathbf{Z};$$

$$\mathbf{\Psi} \equiv \left\{\varphi_j^{(i)}(x)\right\}_{i,j\in\mathbf{Z}}; \quad \mathbf{\Psi}^{(i)} \equiv \left\{\psi_j^{(i)}(x)\right\}_{j\in\mathbf{Z}}, \quad \in \mathbf{Z}.$$

Пример. Порождающие скейлинг-функции и вейвлет ортогонального преобразования Хаара имеют вид:

$$\varphi_H(x) = \begin{cases} 1, & x \in [0,1) \\ 0, & x \notin [0,1) \end{cases}, \quad \psi_H(x) = \begin{cases} 1, & x \in [0,1/2) \\ -1, & x \in [1/2,1) \\ 0, & x \notin [0,1) \end{cases}, \tag{A.6}$$

коэффициенты соответствующих масштабных соотношений:

$$h_0 = \frac{1}{\sqrt{2}}, \quad h_1 = \frac{1}{\sqrt{2}}, \quad g_0 = \frac{1}{\sqrt{2}}, \quad g_1 = -\frac{1}{\sqrt{2}}.$$

A.2 Неортогональный многомасштабный анализ

Требование ортогональности вейвлет-базиса на практике оказывается достаточно сильным ограничением, и его приходится ослаблять.

В определении многомасштабного анализа (п. A.1) требование ортогональности системы базисных скейлинг-функций $\mathbf{\Phi}^{(0)}$ подпространства $V^{(0)}$ можно ослабить и потребовать, чтобы система являлась *базисом Рисса*[2].

Как следствие, базис скейлинг-функций $\mathbf{\Phi}^{(i)}$ в любом подпространстве $V^{(i)}$, $i \in \mathbf{Z}$, также будет базисом Рисса. Любое подпространство $V^{(i+1)}$, $i \in \mathbf{Z}$, представимо в виде объединения подпространств $V^{(i)}$ и $W^{(i)}$, но они не обязаны быть ортогональными друг другу. Из этого, в частности, следует, что по многомасштабному анализу $\mathbf{V}$ соответствующая последовательность детализирующих подпространств $\mathbf{W}$ может определяться неоднозначно.

Базисы вейвлетов $\mathbf{\Psi}^{(i)}$ в детализирующих подпространствах $W^{(i)}$, $i \in \mathbf{Z}$, также должны быть базисами Рисса.

Рассмотрим два неортогональных многомасштабных анализа $\mathbf{V}$ и $\widetilde{\mathbf{V}}$, а также два соответствующих набора детализирующих подпространств $\mathbf{W}$ и $\widetilde{\mathbf{W}}$, таких что:

$$\widetilde{V}^{(0)} \perp W^{(0)}, \quad V^{(0)} \perp \widetilde{W}^{(0)},$$

[2]Последовательность $\{y_j\}$ в гильбертовом пространстве является *базисом Рисса*, если любой элемент y этого пространства может быть представлен единственным образом в виде разложения $y = \sum_j \alpha_j y_j$, и существуют константы $0 < A \le B$, такие что $A\|y\|^2 \le \sum_j |\alpha_j|^2 \le B\|y\|^2$. Очевидно, что для ортонормированного базиса $A = B = 1$ и неравенство превращается в равенство Парсеваля.

а базисные функции $\mathbf{\Phi}^{(0)}$ и $\mathbf{\Psi}^{(0)}$ пространств $V^{(0)}$ и $W^{(0)}$ составляют биортогональные пары с базисными функциями $\widetilde{\mathbf{\Phi}}^{(0)}$ и $\widetilde{\mathbf{\Psi}}^{(0)}$ пространств $\widetilde{V}^{(0)}$ и $\widetilde{W}^{(0)}$ соответственно. Заметим, что если эти требования выполнены для уровня разрешения 0, то они будут выполнены и для любого другого разрешения $i \in \mathbf{Z}$.

При таких условиях вейвлет-базисы $\mathbf{\Psi}$ и $\widetilde{\mathbf{\Psi}}$ пространства $\mathbf{L}_2(\mathbf{R})$ образуют биортогональную пару. (Это же утверждение справедливо и для комбинированных базисов, составленных из скейлинг-функций произвольного уровня разрешения i_0 и всех вейвлетов разрешения, не меньшего, чем i_0).

Выпишем формулы *биортогонального диадного вейвлет-преобразования.*

Прямое преобразование:

$$w_j^{(i)} = \left\langle f(\bullet) \;\middle|\; \tilde{\psi}_j^{(i)}(\bullet) \right\rangle, \quad i,\, j \in \mathbf{Z}. \tag{A.7}$$

Обратное преобразование:

$$f(x) = \sum_{i=-\infty}^{+\infty} \sum_{j=-\infty}^{+\infty} w_j^{(i)}\, \psi_j^{(i)}(x) \tag{A.8}$$

(совпадает с формулой (A.3) для ортогонального случая).

Ортогональное преобразование является частным случаем биортогонального. Действительно, ортонормированный базис биортогонален самому себе и формула (A.7) для такого базиса превращается в уже известную формулу (A.4).

Так же как и для ортогонального случая, возможно разложение сигнала по комбинированному базису, т.е. для любого уровня разрешения $i_0 \in \mathbf{Z}$ формуле (A.8) эквивалентно следующее представление (аналогично (A.5)):

$$f(x) = \sum_{j=-\infty}^{+\infty} v_j^{(i_0)}\, \varphi_j^{(i_0)}(x) + \sum_{i=i_0}^{+\infty} \sum_{j=-\infty}^{+\infty} w_j^{(i)}\, \psi_j^{(i)}(x), \quad i_0 \in \mathbf{Z}, \tag{A.9}$$

где

$$v_j^{(i_0)} = \left\langle f(\bullet) \;\middle|\; \tilde{\varphi}_j^{(i_0)}(\bullet) \right\rangle, \quad i_0,\, j \in \mathbf{Z}.$$

A.3 Вычисление вейвлет-преобразований

Как уже отмечалось в гл. 1, вычисление вейвлет-коэффициентов непосредственно по формулам (A.4) или (A.7) неэффективно.

Избежать таких вычислений помогут соотношения для коэффициентов при базисных функциях соседних уровней разрешения:

$$\begin{aligned} v_j^{(i)} &= \sum_{k=-\infty}^{+\infty} v_{2j+k}^{(i+1)}\, \tilde{h}_k, \\ w_j^{(i)} &= \sum_{l=-\infty}^{+\infty} v_{2j+l}^{(i+1)}\, \tilde{g}_l, \\ & \quad i,\, j \in \mathbf{Z}, \end{aligned} \tag{A.10}$$

$$v_j^{(i+1)} = \sum_{k=-\infty}^{+\infty} \left(v_k^{(i)}\, h_{j-2k} + w_k^{(i)}\, g_{j-2k} \right), \quad i, j \in \mathbf{Z}, \tag{A.11}$$

где $\{\tilde{h}_k\}$, $\{\tilde{g}_k\}$, $\{h_k\}$ и $\{g_k\}$ являются коэффициентами масштабных соотношений для функций $\tilde{\varphi}(x)$, $\tilde{\psi}(x)$, $\varphi(x)$ и $\psi(x)$ соответственно (вывод формул см. в [13, 45]).

Таким образом, если известно представление сигнала с некоторым разрешением i_1, то по формулам (A.10) можно получить представление сигнала с любым разрешением, меньшим i_1, а по формуле (A.11) восстановить исходное представление.

В качестве начального представления с разрешением i_1 берется некоторым образом оцифрованный (дискретизированный) сигнал (получение такой оцифровки не является предметом рассмотрения настоящей работы).

Если же сигнал изначально дискретный ($\mathbf{s} = \{s_j\}_{j\in\mathbf{Z}}$), то можно считать, что он сам и является собственным представлением с разрешением i_1, то есть $v_j^{(i_1)} = s_j,\ j \in \mathbf{Z}$.

Остается заметить, что приведенные в гл. 1 формулы прямого и обратного вейвлет-преобразований (1.5) и (1.6) являются несколько иной формой записи формул (A.10) и (A.11), а используемые в них фильтры состоят из коэффициентов соответствующих масштабных соотношений, а именно:

$$\tilde{\mathbf{h}} = \{\tilde{h}_{-k}\},\ \ \tilde{\mathbf{g}} = \{\tilde{g}_{-k}\};$$
$$\mathbf{h} = \{h_k\},\quad \mathbf{g} = \{g_k\}$$

(для фильтров анализа коэффициенты берутся в обратном порядке).

A.4 Двумерные преобразования

Простейшим примером многомерных преобразований является «естественное» расширение одномерного случая на случай большей размерности. Функциями такого преобразования являются *тензорные произведения* одномерных функций по размерности преобразования. Так для двумерного случая получается четыре порождающих функции — одна скейлинг-функция

$$\varphi\varphi(x,y) \ = \ \varphi(x)\,\varphi(y) \tag{A.12}$$

и три вейвлета

$$\begin{aligned}
\varphi\psi(x,y) &= \varphi(x)\,\psi(y),\\
\psi\varphi(x,y) &= \psi(x)\,\varphi(y),\\
\psi\psi(x,y) &= \psi(x)\,\psi(y).
\end{aligned} \tag{A.13}$$

(для наглядности мы будем придерживаться несколько нестандартных двухбуквенных обозначений, что, во-первых, позволит избежать введения новых символов, а, во-вторых, отражает структуру данного вида преобразований).

Все остальные функции определяются соотношением:

$$\Omega_{j,k}^{(i)}(x,y) = 2^i \Omega(2^i x - j, 2^i y - k), \quad i,j \in \mathbf{Z}, \tag{A.14}$$

где символ Ω заменяется на $\varphi\varphi$, $\varphi\psi$, $\psi\varphi$ или $\psi\psi$.

Если система, порожденная функциями (A.13), является ОНБ в $\mathbf{L}_2(\mathbf{R}^2)$ (для чего необходимо и достаточно, чтобы система, порожденная функциями $\varphi(x)$ и $\psi(x)$ являлась ОНБ в $\mathbf{L}_2(\mathbf{R})$), то прямое вейвлет-преобразование сигнала $f(x,y) \in \mathbf{L}_2(\mathbf{R}^2)$ будет вычисляться по формуле:

$$\begin{aligned} vw_{j,k}^{(i)} &= \left\langle f(\bullet) \;\middle|\; \varphi\psi_{j,k}^{(i)}(\bullet) \right\rangle; \\ wv_{j,k}^{(i)} &= \left\langle f(\bullet) \;\middle|\; \psi\varphi_{j,k}^{(i)}(\bullet) \right\rangle; \\ ww_{j,k}^{(i)} &= \left\langle f(\bullet) \;\middle|\; \psi\psi_{j,k}^{(i)}(\bullet) \right\rangle; \\ & i,j,k \in \mathbf{Z}, \end{aligned} \tag{A.15}$$

а обратное:

$$\begin{aligned} f(x,y) = \sum_{i=-\infty}^{+\infty} \sum_{j=-\infty}^{+\infty} \sum_{k=-\infty}^{+\infty} \Big(& vw_{j,k}^{(i)}\, \varphi\psi_{j,k}^{(i)}(x,y) + \\ & + wv_{j,k}^{(i)}\, \psi\varphi_{j,k}^{(i)}(x,y) + ww_{j,k}^{(i)}\, \psi\psi_{j,k}^{(i)}(x,y) \Big). \end{aligned} \tag{A.16}$$

Если же одномерный базис не ортогонален и имеет биортогональную пару, порожденную функциями $\tilde{\varphi}(x)$ и $\tilde{\psi}(x)$, то соответствующие двумерные базисы, полученные с помощью тензорного произведения также будут биортогональны. Обобщить формулы (A.15) и (A.16) на биортогональный случай не составляет труда.

В двумерном случае также возможно разложение сигнала по комбинированному базису (т.е. базису, содержащему скейлинг-функции $\varphi\varphi_{j,k}^{(i_0)}(x,y)$, $j,k \in \mathbf{Z}$, некоторого уровня разрешения $i_0 \in \mathbf{Z}$):

$$\begin{aligned} f(x,y) = &\sum_{j=-\infty}^{+\infty}\sum_{k=-\infty}^{+\infty} vv_{j,k}^{(i_0)}\,\varphi\varphi_{j,k}^{(i_0)}(x,y) + \\ &+ \sum_{i=i_0}^{+\infty}\sum_{j=-\infty}^{+\infty}\sum_{k=-\infty}^{+\infty}\Big(vw_{j,k}^{(i)}\,\varphi\psi_{j,k}^{(i)}(x,y) + \\ &+ wv_{j,k}^{(i)}\,\psi\varphi_{j,k}^{(i)}(x,y) + ww_{j,k}^{(i)}\,\psi\psi_{j,k}^{(i)}(x,y)\Big), \end{aligned} \tag{A.17}$$

где (в ортогональном случае):

$$vv_{j,k}^{(i_0)} = \left\langle f(\bullet) \;\middle|\; \varphi\varphi_{j,k}^{(i_0)}(\bullet)\right\rangle,$$

$$j,k \in \mathbf{Z}.$$

Вычисление коэффициентов двумерного преобразования не требует вывода новых формул, т.к. сводится к композиции шагов одномерных преобразований (гл. 1, п. 1.5).

A.5 Нормализация вейвлет-базисов

Часто в приложениях требуется разложение сигналов не по нормированному базису, а по базису функций, имеющих, например, одинаковые максимальные и минимальные значения (условно назовем такой базис ненормализованным). В таком случае из масштабных соотношений следует убрать нормирующий множитель $\sqrt{2}$, что повлечет за собой изменение коэффициентов этих соотношений (и, следовательно, фильтров). Так, в случае нормализованного базиса сумма коэффициентов НЧ фильтров (и синтеза, и анализа) должна быть равна $\sqrt{2}$, а ненормализованного 1 для НЧ анализа и 2 для НЧ синтеза. Любой вейвлет-базис можно либо нормализовывать, либо нет.

Фильтры преобразования Хаара, приведенные в гл. 1, а также фильтры B-сплайнового преобразования (гл. 3) соответствуют ненормализованному случаю, фильтры преобразования D_4 (гл. 1) — нормализованному.

Подробнее о нормализации преобразований см. [13].

Список литературы

1. Баяковский Ю.М., Галактионов В.А., Михайлова Т.Н. Графор. Графическое расширение Фортрана. — М.: Наука, 1985.

2. Бердышев В.И., Петрак Л.В. Аппроксимация функций, сжатие численной информации, приложения. — Екатеринбург: УрО РАН, 1999.

3. Добеши И. Десять лекций по вейвлетам. — Ижевск: НИЦ «Регулярная и хаотическая динамика», 2001.

4. Захаров В.Г. Разработка и применение методов вейвлет-анализа к нелинейным гидродинамическим системам: Дис. . . . канд. физ.-мат. наук: 01.02.05 — Пермь, 1997.

5. Иванов В.П., Батраков А.С. Трехмерная компьютерная графика. — М.: Радио и связь, 1995.

6. Каханер Д., Моулер К., Нэш С. Численные методы и программное обеспечение. — М.: Мир, 2001.

7. Кирушев В.А. Быстрый алгоритм сжатия изображнений // Вестник молодых ученых. Прикладная математика и механика. — 1997. — № 1. — С. 4-10.

8. Кокорин О.Ю., Упольников С.А. Использование иерархического алгоритма в методе излучательности // Труды конференции ГрафиКон'97. — 1997. — С. 31-37.

9. Левкович-Маслюк Л.И. Аппроксимация финансовых рядов фрактальными интерполяционными функциями // Вопросы анализа риска. — 1998. — Vol. 1, No. 1.

10. Лоренц Р.А., Саакян А.А. О подпространствах, порожденных всплеск-системами // Математические заметки. — 1998. — Т. 63, № 2. — С. 299-302.

11. Новиков И.Я., Стечкин С.Б. Основные конструкции всплесков // Фундаментальная и прикладная математика. — 1997. — Т. 3, № 4. — С. 999-1028.

12. Новиков И.Я., Стечкин С.Б. Основные теории всплесков // Успехи математических наук. — 1998. — Т. 53, № 6. — С. 53-128.

13. Переберин А.В. О систематизации вейвлет-преобразований // Вычислительные методы и программирование. — 2001. — Т. 2, № 2. — С. 133-158. (Электронная версия на сайте http://num-meth.srcc.msu.su/)

14. Переберин А.В. Построение изолиний с автоматическим масштабированием // Вычислительные методы и программирование. — 2001. — Т. 2, № 1. — С. 118-128. (Электронная версия на сайте http://num-meth.srcc.msu.su/)

15. Петухов А.П. Введение в теорию базисов всплесков. — СПб.: Изд-во СПбГТУ, 1999.

16. Поспелов В.В., Кислицына М.А. Использование преобразования Хаара для модификации алоритма JPEG сжатия изображений // Тезисы докладов коференции РОАИ'97. — 1997. — Ч. 1. — С. 210-212.

17. Роджерс Д., Адамс Дж. Математические основы машинной графики. — М.: Мир, 2001.

18. Стаховский И.Р. Вейвлетный анализ временных сейсмических рядов // ДАН. — 1996. — Т. 350, № 3. — С. 393-396.

19. Шикин Е.В., Боресков А.В. Компьютерная графика. Динамика, реалистические изображения. — М.: ДИАЛОГ-МИФИ, 1996.

20. Чуи К. Введение в вэйвлеты. — М.: Мир, 2001.

21. De Bonet J.S. Multiresolution Sampling Procedure for Analysis and Synthesis of Texture Images // SIGGRAPH'97 Proceedings. — 1997. — P. 362-368.

22. Bonneau G.-P. Multiresolution Analysis of Irregular Surface Meshes // IEEE Trans. on Visualization and Computer Graphics. — 1998. — Vol. 4, No. 4. — P. 365-378.

23. Bonneau G.-P., Hahmann S., Nielson G.M. BLaC-Wavelets: A Multiresolution Analysis With Non-Nested Spaces // IEEE Visualization'96 Proceedings. — 1996. — P. 43-48.

24. Burt P.J., Adelson E.H. The Laplasian Pyramid as a Compact Image Code // IEEE Trans. on Communications. — 1983. — Vol. COM-31, No. 4. — P. 532-540.

25. Chui C.K. An Introduction to Wavelets. — New York - London: Academic Press, 1992.

26. Chui C.K., editor. Wavelets: a Tutorial in Theory and Applications. — New York - London: Academic Press, 1992.

27. Chui C.K., Quak E. Wavelets on a Bounded Interval // Numerical Methods in Approximation Theory. — 1992. — Vol 9. — P. 53-75.

28. Coifman R.R., Meyer Y., Wickerhauser V. Wavelet Analysis and Signal Processing Wavelets // Wavelets and their Applications. — Boston: Jones and Barlett, 1992. — P. 153-178.

29. Daubechies I. Ten Lectures on Wavelets. — Philadelphia: SIAM, 1992.

30. Daubechies I., Sweldens W. Factoring Wavelet Transforms into Lifting Steps // IEEE Trans. on Image Processing. — 2000. — Vol. 9, No. 3. — P. 480-496.

31. DeVore R., Jawerth W., Lucier B. Image Compression Trough Wavelet Transform Coding // IEEE Trans. on Information Theory. — 1992. — Vol. 39, No. 2. — P. 719-746.

32. Eck M., DeRose T.D., Duchamp T., Hoppe H., Lounsbery M., Stuetzle W. Multiresolution Analysis of Arbitrary Meshes // SIGGRAPH'95 Proceedings. — 1995. — P. 173-182.

33. Efros A.A., Leung T.K. Texture Synthesis by Non-Parametric Sampling // IEEE International Conference on Computer Vision. 1999.

34. Finkelstein A., Salesin D.H. Multiresolution Curves // SIGGRAPH'94 Proceedings. — 1994. — P. 261-268.

35. Foley J.D., van Dam A., Feiner S.K., Huges J.F. Computer Graphics. Principles and Practice. — New York: Addison-Wesley, 1990.

36. Gabour D. Theory of Communications // Journal of the Institute of Electrical Engineers. — 1946. — Vol. 93, No. 22. — P. 429-457.

37. Glassner A.S. Principles of Digital Image Synthesis. — San Francisco: Morgan Kaufmann, 1995.

38. Gortler S.J., Schröder P., Cohen M.F., Hanrahan P. Wavelet Radiosity // SIGGRAPH'93 Proceedings. — 1993. — P. 221-230.

39. Gross M.H., Staadt O.G., Gatti R. Efficient Triangular Surface Approximations Using Wavelets and Quadtree Data Structures // IEEE Trans. on Visualization and Computer Graphics. — 1996. — Vol. 2, No. 2. — P. 130-143.

40. Haar A. Zur Theorie der Orthogonalen Funktionen-Systeme // Math. Ann. — 1910. — No. 69. — P. 331-371.

41. Heeger D.J., Bergen J.R. Pyramid-based Texture Analysis/Synthesis // SIGGRAPH'95 Proceedings. — 1995. — P. 229-238.

42. Ihm I., Park S. Wavelet-Based 3D Compression Scheme for Ineractive Visualization of Very Large Volume Data // Computer Graphics Forum. — 1999. — Vol. 18, No. 1. — P. 3-15.

43. Ivanov D.V., Kuzmin E.P., Burtsev S.V. Progressive Image Compression Using Binary Trees // GraphiCon'98 Proceedings. — 1998. — P. 187-194.

44. Jacobs C.E., Finkelstein A., Salesin D. Fast Multiresolution Image Querying // SIGGRAPH'92 Proceedings. — 1992. — P. 177-184.

45. Jawerth B., Sweldens W. An Overwiew of Wavelet Based Multiresolution Analyses // SIAM Rev. — 1994. — Vol. 36, No. 3. — P. 377-412.

46. Jensen H.W. Global Illumination Using Photon Maps // The 7-th Eurographics Workshop on Rendering Proseedings. — 1996. — P. 21-30.

47. Jensen H.W., Christensen N.J. Bidirectional Monte Carlo Ray Tracing of Copmplex Objects Using Photon Maps // Computers and Graphics. — 1995. — Vol. 19, No. 2.

48. Kortchagine D.N., Krylov A.S. Projection Filtering in Image Processing // GraphiCon'2000 Proceedings. — 2000. — P. 42-45.

49. Kovačević J., Sweldens W. Wavelet Families of Increasing Order in Arbitrary Dimentions // IEEE Trans. on Image Processing. — 2000. — Vol. 9, No. 3. — P. 480-496.

50. Lee S., Lawton W., Shen Z. An Algorithm for Matrix Extension and Wavelet Constructions // Mathematics of Computation. — 1996. — Vol. 65, No. 214. — P. 723-737.

51. Levkovich-Maslyuk L.I. Wavelet-Based Determination of Generating Matrices for Fractal Interpolation Functions // Regular and Chaostic Dynamics. — 1998. — Vol. 3, No. 2.

52. Lorensen W.E., Cline H.E. Marching Cubes: A High Resolution 3D Surface Reconstruction Algorithm // Computer Graphics. — 1987. — Vol. 21, No. 4. — P. 163-169.

53. Lounsbery M. Multiresolution Analysis for Surfaces of Arbitrary Topological Type: PhD thesis / Univ. of Washington. — Seattle, 1994.

54. Lounsbery M., T.D. DeRose T.D., Warren J. Multiresolution Analysis for Surfaces of Arbitrary Topological Type // ACM Trans. on Graphics. — 1997. — Vol. 16, No. 1. — P. 34-73.

55. Mallat S. Multiresolution Approximation and Wavelet Othonormal Bases $\mathbf{L}_2(\mathbf{R})$ // Trans. AMS. — 1989. — Vol. 1, No. 315. — P. 69-87.

56. Mallat S. A Wavelet Tour of Signal Processing. — New York - London: Academic Press, 1998.

57. Mallat S., Zhong S. Characterization of Signals from Multiscale Edges // IEEE Trans. on Pattern Analysis and Machine Intelligence. — 1992. — Vol. 14, No. 7. — P. 710-732.

58. Martens J.-B. The Hermite Transform — Theory // IEEE Trans. on Acoustics, Speech and Signal Processing. — 1990. — No. 38. — P. 1595-1606.

59. Martens J.-B. The Hermite Transform — Applications // IEEE Trans. on Acoustics, Speech and Signal Processing. — 1990. — No. 38. — P. 1607-1618.

60. Myszkowski K. Lighting Reconstruction Using Fast and Adaptive Density Estimation Techniques // Rendering Techniques'97. 1997.

61. Mumford D., Gidas B. Stochastic Models for Generic Images. — 2000. (http://www.dam.brown.edu/people/mumford/Papers/Generic5.pdf)

62. Pereberin A.V. From Photon Map to Irradiance Function via Wavelet Transform // GraphiCon'97 Proceedings. — 1997. — P. 38-43.

63. Pereberin A.V. Hierarchical Approach for Texture Compression // GraphiCon'99 Proceedings. — 1999. — P. 195-199.

64. Pereberin A.V. Fast Multi-Scaled Texture Generation and Rendering // GraphiCon'2000 Proceedings. — 2000. — P. 145-150.

65. Rogers D. Introduction to NURBS: With Historical Perspective. — San Francisco: Morgan Kaufmann, 2000.

66. Said A., Perlman W.A. A New Fast and Efficient Image Codec Based on Set Partitioning in Hierarchical Trees // IEEE Trans. CSVT. — 1996. — Vol. 6, No. 3. — P. 243-250.

67. Schroeder W.J., Zarge J.A., Lorensen W.E. Decimation of Triangle Meshes // SIGGRAPH'92 Proceedings. — 1992. — P. 65-70.

68. Shapiro J.M. Embedded Image Coding Using Zerotrees of Wavelet Coefficients // IEEE Trans. on Signal Processing. — 1993. — Vol. 41, No. 12. — P. 345-362.

69. Shirley P. Time Complexity of Monte Carlo Radiosity // Eurographics'91 Proceedings. — 1991. — P. 459-466.

70. Shirley P., Wade B., Hubbard P.M., Zareski D., Walter B., Greenberg D.P. Global Illumination via Density Estimation // The 6-th Eurographics Workshop on Rendering Proseedings. — 1995. — P. 219-230.

71. Silverman B.W. Density Estimation for Statistics and Data Analysis. — London: Chapmann and Hall, 1985.

72. Smits B.E., Arvo J.R., Greenberg D.P. A Clustering Algorithm for Radiosity in Complex Envirounments // SIGGRAPH'94 Proceedings. — 1994. — P. 435-442.

73. Stollnitz E.J., DeRose T.D., Salesin D.H. Wavelets for Computer Graphics. Theory and applications. — San Francisco: Morgan Kaufmann, 1996.

74. Sweldens W. The Lifting Scheme: A Construction of Second Generation Wavelets // SIAM J. Math. Anal. — 1996. — Vol. 3, No. 2. — P. 186-200.

75. Sweldens W., Schröder P. Building Your Own Wavelets at Home // Wavelets in Computer Graphics, ACM SIGGRAPH Course Notes, 1996.

76. Tieng Q.M., Boles W.W. Recognition of 2D Object Contours Using the Wavelet Transform Zero-Crossing Representation // IEEE Trans. on Pattern Analysis and Machine Intellegence. — 1997. — Vol. 19, No. 8. — P. 910-916.

77. Wojtaszczyk P. A Mathematical Introduction to Wavelets. — Cambridge: Cambridge University Press, 1997.

78. Zakharov V. Nonseparable Multidimentional Littlewood-Paley Like Wavelet Bases. — Marseille: Centre de Phisique Théorique, 1996.