СОДЕРЖАНИЕ

Номер 3, 2020

АНАЛИЗ ДАННЫХ

КОМПЬЮТЕРНАЯ ГРАФИКА И ВИЗУАЛИЗАЦИЯ

CONTENTS

No. 3 2020

DATA ANALYSIS

COMPUTER GRAPHICS AND VISUALIZATION

ПРОГРАММИРОВАНИЕ, 2020, № 3, с. 3–15

АНАЛИЗ ДАННЫХ

УДК 004.932

СЕГМЕНТАЦИЯ НА ОСНОВЕ РАСПРОСТРАНЕНИЯ ДИНАМИЧЕСКИ ИЗМЕНЯЕМЫХ СУПЕРПИКСЕЛЕЙ

© 2020 г. В. В. Данилов[a,*], О. М. Гергет[a,b,], И. П. Скирневский[a], Р. А. Манаков[a], Д. Ю. Колпащиков[a]**

[a] *Томский политехнический университет, Лаборатория дизайна медицинских изделий 634050 Томск, проспект Ленина, д. 2, стр. 33, Россия*

[b] *Сибирский государственный медицинский университет 634050 Томск, Московский тракт, д. 2, Россия*

**E-mail: viacheslav.v.danilov@gmail.com*

***E-mail: olgagerget@mail.ru*

Поступила в редакцию 19.12.2019 г.
После доработки 20.01.2020 г.
Принята к публикации 20.01.2020 г.

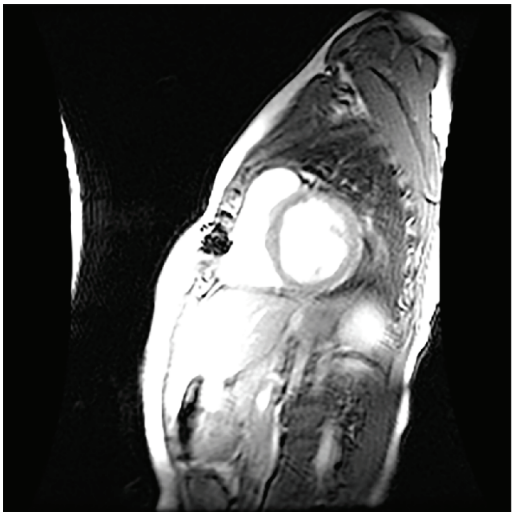
Представленная работа посвящена новому методу сегментации медицинских данных на основе распространения суперпикселей. Предлагаемый метод является модификацией классического алгоритма сегментации на основе выращивания регионов и отчасти наследует концепцию октодеревьев. Основным отличием данного подхода является переход в домен суперпикселей, а также более гибкие условия объединения суперпикселей в регион. В ходе выполнения алгоритма для объединения суперпикселей выполняются проверки на соответствие критериям однородности. Во-первых, средняя интенсивность суперпикселя сравнивается с интенсивностью полученного региона. Во-вторых, каждый пиксель, лежащий на ребрах и диагоналях суперпикселя, сравнивается с пороговым значением. Важной особенностью предлагаемого метода является динамически изменяемый (плавающий) размер суперпикселей. Формирование итогового региона осуществляется посредством построения сплайна по точкам пересечения внешних, по отношению к региону, суперпикселей. В качестве данных для оценки точности метода использовались МРТ изображения левого желудочка сердца, полученные в Университете Йорка, и опухоли головного мозга, полученные в Южном медицинском университете. С целью демонстрации быстродействия метода дополнительно создан набор синтетических изображений высокого разрешения. В качестве метрики оценки точности использован коэффициент подобия Дайса. Его значения для предлагаемого метода соответствуют 0.93 ± 0.03 и 0.89 ± 0.07 при сегментации левого желудочка и опухоли соответственно. На основе приведенного в статье примера показано, что поэтапное уменьшение размера суперпикселя позволяет значительно увеличить скорость работы метода без потери точности.

DOI: 10.31857/S0132347420030048

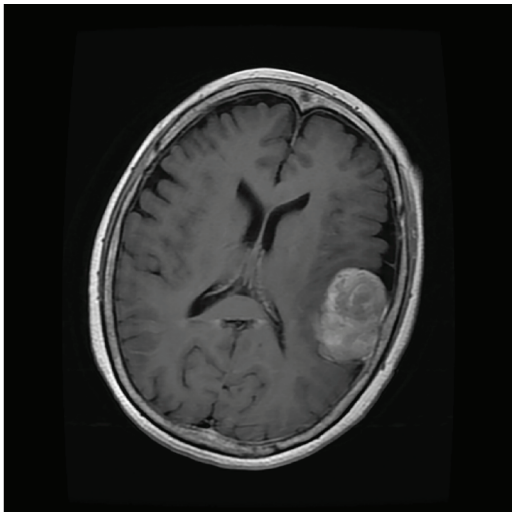
1. ВВЕДЕНИЕ

Одной из основных задач при работе с медицинскими изображениями является точное выделение области интереса. Данная задача может быть решена методами сегментации [1–3]. Однако, для получения результатов, удовлетворяющих заданной точности, требуется проведение предобработки исходных данных. Следует отметить, что полуавтоматические методы сегментации все еще остаются популярными, поскольку не предъявляют высоких требований к вычислительным ресурсам и являются относительно простыми в реализации. В настоящее время хорошо зарекомендовали себя в области обработки медицинских данных методы машинного обучения, так как позволяют получить высокую точность распознавания образов и устойчивость к различного рода шумам и артефактам [4–6]. Машинное обучение активно применяется для решения задач выделения новообразований, в частности, с целью сегментации опухолей головного мозга [7]. Активное внедрение нейронных сетей наблюдается не только для сегментации изображений. Спектр применения методов этого класса достаточно широк: классификации, кластеризация, поиск объектов, шумоподавление и другое [8, 9].

Несмотря на то, что методы на основе нейронных сетей позволяют выполнять сегментацию трехмерных объектов, популярными остаются подходы, при которых трехмерные данные раскладываются на массивы двумерных проекций так называемых слайсов. Каждый слайс обраба-



(а) МРТ сердца (набор данных Университета Йорка)



(б) МРТ головного мозга (набор данных Южного медицинского университета)

Рис. 1. Примеры входных данных для оценки предлагаемого метода сегментации.

тывается отдельно, а затем происходит процесс восстановления 3D объекта [10, 11]. Разложение трехмерных данных на двумерные составляющие является эффективным инструментом и активно используется в медицине, поскольку в большинстве случаев не требуется проведения анализа всего анатомического объекта. Подобные упрощения играют важную роль в развитии классических методов сегментации, к числу которых относятся методы активных контуров (AC, сокр. от Active Contours), методы выращивания регионов (RG, сокр. от Region Growing), водоразделов (от англ. Watershed). Следует также отметить, что во многих программных продуктах, используемых для разметки и подготовки данных, применяются встроенные полуавтоматические методы сегментации, которые позволяют оператору значительно упростить процесс разметки.

В данной работе продемонстрирован метод двухмерной сегментации медицинских данных на основе распространения суперпикселей с динамически изменяющимся размером. Концепция данного метода базируется на работах Dana Ballard и Richard Adams [12, 13]. Ключевая особенность метода при делении суперпикселей была унаследована от октодеревьев.

2. СВЯЗАННЫЕ ИССЛЕДОВАНИЯ

Метод сегментации на основе выращивания регионов был впервые представлен более десяти лет назад, и постоянно ведутся исследования с целью его улучшения. В работе [14] Jun Tang предлагает выполнять сегментацию цветных изображений на основе комбинации методов выращивания регионов и водоразделов. Несмотря на то, что совместная реализация двух методов дала прирост в точности распознавания, сами методы по-прежнему остаются трудоемкими.

Одна из известных модификаций метода выращивания регионов представлена в работе Joung Park и Chulhee Lee [15]. В данной работе применение морфологических алгоритмов позволило избежать ручной установки начальной точки, которая является обязательным шагом в классическом варианте метода. На основе морфологической маски автоматически осуществлялся поиск начальной точки области интереса и заднего фона. Подобные улучшения применялись и другими авторами, позволяя перейти из домена полуавтоматической сегментации к автоматическим методам. Однако такие известные проблемы этих методов как пересегментация и чувствительность к шуму не были устранены до сих пор [16, 17].

С целью автоматического вычисления начальной точки для метода выращивания регионов были предложены различные подходы, которые позволяли с разной точностью определить пиксели, связанные с областью интереса без участия оператора. В работе [18] Nor Ashidi Mat Isa предлагает целую серию изменений метода RG, связанных с определением порогового значения, модернизацией процесса присоединения пикселей к региону интереса и инициализацией алгоритма. В частности, в статье описывается использование метода кластеризации k-средних для определения области интереса и заднего фона. Все описанные улучшения значительно увеличивают точность

Рис. 2. Алгоритм сегментации на основании распространения суперпикселей.

метода, однако обладают высокой вычислительной сложностью, что сильно сказывается на времени сегментации.

В процессе исследований также были предприняты попытки комбинирования метода RG и нечеткой логики. В работе [19] авторы применяют так называемый нечетко связанный метод выращивания регионов (Fuzzy Connectivity Region Growing) совместно с классическим методом выращивания регионов.

Другим направлением в исследованиях по сегментации изображений является переход в так называемый домен суперпикселей [20–23]. Одно из самых ранних упоминаний понятия суперпикселя с подробным описанием сути метода встречается в работе Xiaofeng Ren и Jitendra Malik [24]. Под суперпикселем понимается группа пикселей, объеденных по общему признаку или группе признаков. Основным преимуществом суперпикселей является переход от элементарных составляющих изображений к группам пикселей с дальнейшим рассмотрением группы, как целого объекта. При этом ключевым недостатком большинства методов, основанных на суперпикселях, является обработка каждого пикселя в рамках группы при формировании суперпикселя. Такое ограничение не позволяет получить значительных улучшений в производительности и точности метода. Самым распространенным методом группировки пикселей является линейная кластеризация (Simple Linear Iterative Clustering) [7, 20, 22]. Детальное описание перехода в домен суперпикселей представлено в работе Radhakrishna Achanta [25]. Автор проводит сегментацию на основе суперпикселей с использованием метода k-средних и пятимерного пространства признаков. Подход на основе суперпикселей нашел свое применение и в сегментации изображений с высоким разрешением. В работе [21] Ovidiu Csillik демонстрирует метод сегментации изображений с высоким разрешением на основании суперпикселей, полученных линейной кластеризацией. Время, затраченное на сегментацию изображений разрешением 1347 × 1042 и 3701 × 3301 пикселей, составило соответственно 2 и 26 секунд.

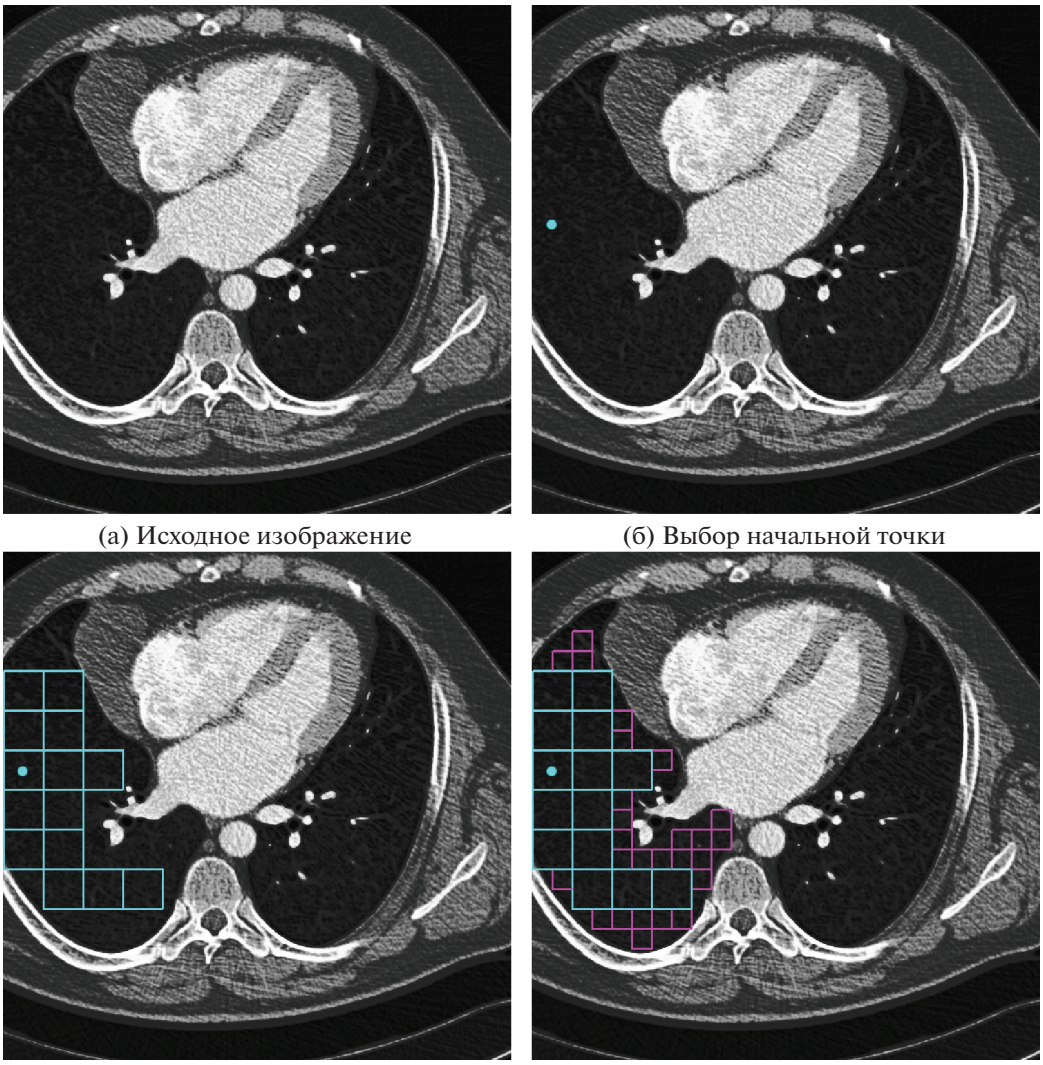
Несмотря на то, что двухмерная сегментация на основании суперпикселей достаточно широко распространена в литературе, в большинстве статей авторы не решают проблем, связанных с высокой сложностью алгоритмов, и, как следствие, длительным временем выполнения сегментации. В данной работе представлен метод, который при сохранении точности позволяет значительно снизить скорость обработки изображений за счет использования суперпикселей с динамически изменяемым размером. При этом формирование суперпикселей происходит без прохода по каждому пикселю изображения.

3. ДАННЫЕ И МЕТОДЫ

3.1. Исходные данные

Для исследования и оценки точности алгоритмов были использованы данные, полученные в Университете Йорка (Йорк, Великобритания) [26] и Южном медицинском университете (Гуанчжоу, Китай) [27, 28]. Эти данные находятся в открытом доступе и могут использоваться для проведения исследований при указании ссылки на источник.

Первый набор данных содержит данные МРТ сердца для 33 пациентов. Данные на одного пациента представляют собой 20 фреймов, каждый из которых состоит из 8–15 слайсов. Общее количество данных составляет 7980 двухмерных изобра-



(а) Исходное изображение

(б) Выбор начальной точки

(в) Распространение суперпикселей начального размера (голубые блоки)

(г) Уменьшение размера и повторное распространение суперпикселей (фиолетовые блоки)

Рис. 3. Концепция распространения суперпикселей внутри области интереса с последующим изменением их размера.

жений с разрешением 256 × 256 пикселей. Второй набор данных содержит трехмерные данные МРТ головного мозга 233 пациентов (в том числе с диагностируемыми менингиомой, глиомой, опухолью гипофиза и состоит из 3064 изображений с разрешением 512 × 512 пикселей. Разметка данных проводилась вручную и была продублирована двумя опытными врачами. На изображениях сердца были получены контуры эпикарда и эндокарда левого желудочка. На данных головного мозга выполнена сегментация новообразований. Примеры изображений приведены на рисунке 1. Все изображения обрабатывались на компьютере, оснащенным Intel Core i7-4820K 3.7 GHz CPU и NVIDIA GeForce GTX 960 с использованием MATLAB.

3.2. Сегментация на основе выращивания регионов

Метод выращивания регионов построен по принципу присоединения соседних пикселей к начальной точке области интереса. Присоединение происходит исходя из условия однородности. В классической реализации алгоритма условием однородности является отклонение по яркости между соседними пикселями. При этом связанными считаются пиксели, которые удовлетворяют критериям 4-связности или 8-связности.

В классической версии алгоритма существует две реализации: с начальной точкой, которая устанавливается вручную внутри области интереса; без начальной точки, в которой установка первого пикселя происходит случайным образом. Упрощенно работу алгоритма сегментации на ос-

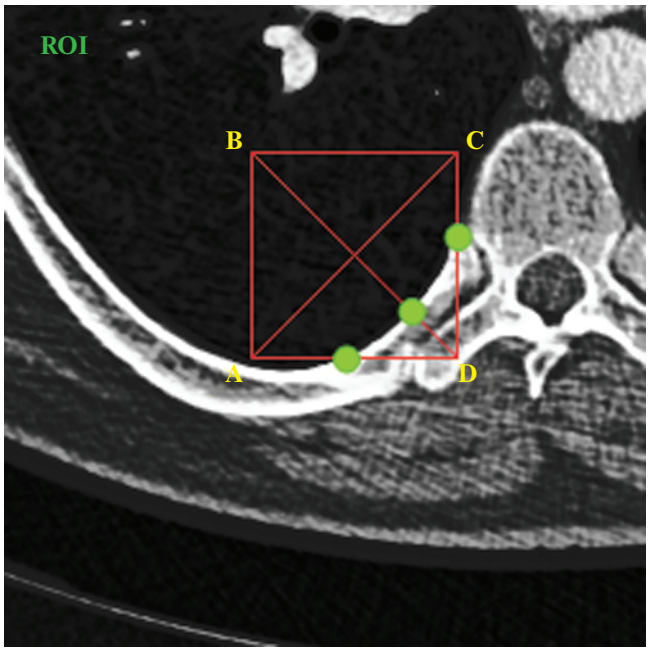


Рис. 4. Суперпиксель (красный блок), пересекающий границу области интереса в трех точках (зеленые точки).

нове выращивания регионов можно разделить на следующие шаги:

1. Выбор начальной точки.

2. Сравнение начальной точки с соседними пикселями по условию однородности.

3. Остановка алгоритма, если все пиксели пройдены.

Функция однородности соседних пикселей описана в работе [13] и представлена ниже:

$$P(x,r) = \left|f(x) - \mu_r\right| < T, \qquad (3.1)$$

где $f(x)$ – яркость текущего пикселя, μ_r – средняя яркость региона, T – пороговое значение.

3.3. Распространение суперпикселей

Как было отмечено ранее, представленный в статье метод сегментации на основании распространения суперпикселей (SPPS, сокр. от Super-Pixel Propagation Segmentation) имеет схожий с методом RG принцип работы. Однако, все вычисления переводятся в домен суперпикселей, что позволяет значительно снизить сложность алгоритма и сократить время работы. В статье используется стандартное определение суперпикселя. Для двухмерного случая суперпикселем считается группа связанных пикселей квадратной формы, объединенных некоторым условием однородности. Пиксели считаются связанными, если удовлетворяют критериям 4-связности. Соединение соседних суперпикселей в общий регион происходит на основании нескольких критериев, описанных в пункте 3.4. Итоговый контур формируется посредством сплайна Эрмита. Предложенный алгоритм сегментации SPPS приведен на рисунке 2.

В отличии от подходов, описанных в работах [22, 29], алгоритм SPPS не анализирует каждый пиксель в составе суперпикселя. Так, при переходе в домен суперпикселей размерностью 10 × 10 пикселей, в стандартных реализациях будут обработаны все 100 пикселей, что соответствует вычислительной сложности равной $O(n^2)$. При размере суперпикселя 10 × 10 пикселей для каждого суперпикселя будет обработано от 30 до 60 пикселей без потери точности сегментации (количество может меняться в зависимости от расположения суперпикселя). Формирование и анализ суперпикселей описаны в пункте 3.4. Концепция распространения суперпикселей с изменяемым размером представлена на рисунке 3.

Аналогично методу выращивания регионов, процесс объединения происходит до тех пор, пока есть хотя бы один суперпиксель, который может быть включен в итоговый регион. Включение суперпикселя происходит только в том случае, если его стороны не пересекают границу области интереса (см. пункт 3.4). Если на ребрах или диагоналях суперпикселя имеется хотя бы одно пересечение, то он не включается в регион, и алгоритм переходит к следующему суперпикселю.

Первый шаг алгоритма заключается в определении центральной точки начального суперпикселя. Выбор точки происходит вручную, таким образом, метод SPPS относится к классу полуавтоматических. После того как выбрана точка, вокруг нее формируется суперпиксель заданного размера. Затем диагонали и ребра суперпикселя проверяются на предмет пересечения с границей области интереса. Если суперпиксель не пересекает границу области интереса, шаг алгоритма считается успешно завершенным. В противном случае алгоритм может продолжить работу только в том случае, если размер исходного суперпикселя будет уменьшен без изменения позиции центральной точки.

3.4. Поиск граничных суперпикселей

В предложенном алгоритме для определения границы области интереса осуществляется обязательная проверка двух условий. Для первого условия, представленного ниже, яркость пикселей, лежащих на ребрах и диагоналях суперпикселя, сравнивается со средней яркостью центральных пикселей уже объединенных суперпикселей. Пороговый параметр является настраиваемым параметром метода.

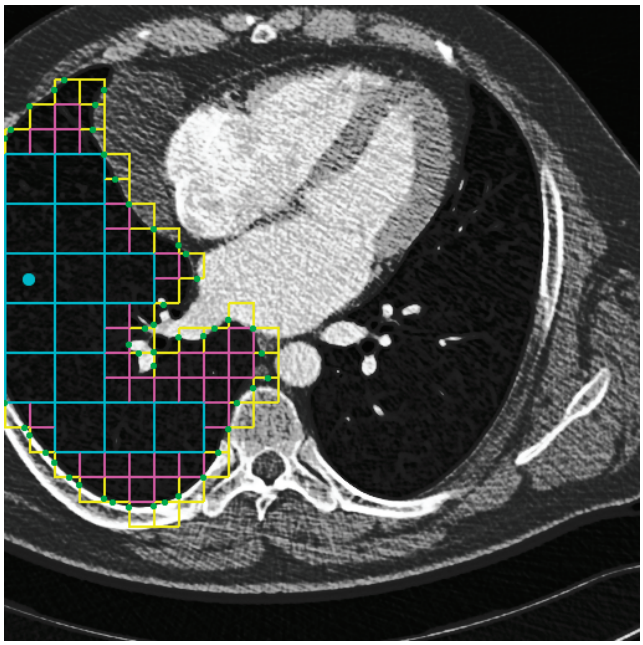


Рис. 5. Пример внешних суперпикселей, которые формируют контур области интереса (желтые блоки). Зеленые точки демонстрируют пересечения суперпикселей с границей объекта.

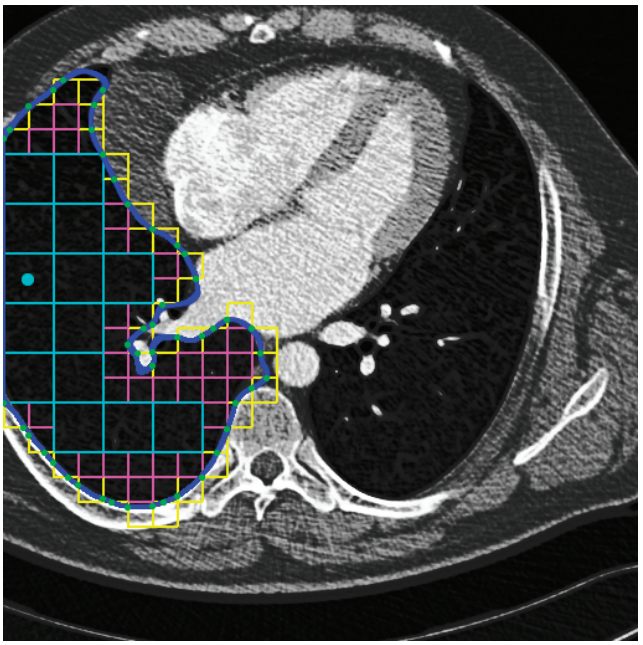


Рис. 6. Кубический сплайн Эрмита (синяя линия), проходящий через узловые точки (зеленые точки).

$$\Delta I = \left| p_i - \frac{\sum_{j=1}^{k} p_{c_j}}{k} \right| \geq T, \qquad (3.2)$$

где p_i – яркость текущего пикселя, p_{c_j} – интенсивность центрального пикселя, присоединенного суперпикселя, k – число присоединенных суперпикселей, T – пороговое значение яркости.

Следующее условие, проверяемое параллельно, является аппроксимацией яркостей прямой. Аппроксимация производится методом наименьших квадратов по 5 точкам. Коэффициент, являющийся тангенсом угла наклона прямой, вычисляется согласно следующей формуле:

$$\tan(\varphi) = \left| \frac{n\sum_{i=1}^{n} x_i y_i - \sum_{i=1}^{n} x_i \sum_{i=1}^{n} y_i}{n\sum_{i=1}^{n} x_i^2 - \left(\sum_{i=1}^{n} x_i\right)^2} \right| \geq Slope, \qquad (3.3)$$

где x – точки, расположенные на ребрах и диагоналях суперпикселя (в данном случае от 1 до 5), y – значение яркости, n – позиция точки на ребре. При построении вектора x следует учитывать, что расстояние между пикселями является Евклидовым.

Для текущего пикселя аппроксимация производится с использованием двух пикселей слева и двух справа. Если пиксели являются граничными для данного суперпикселя, то используются дополнительные пиксели, близлежащие к границе

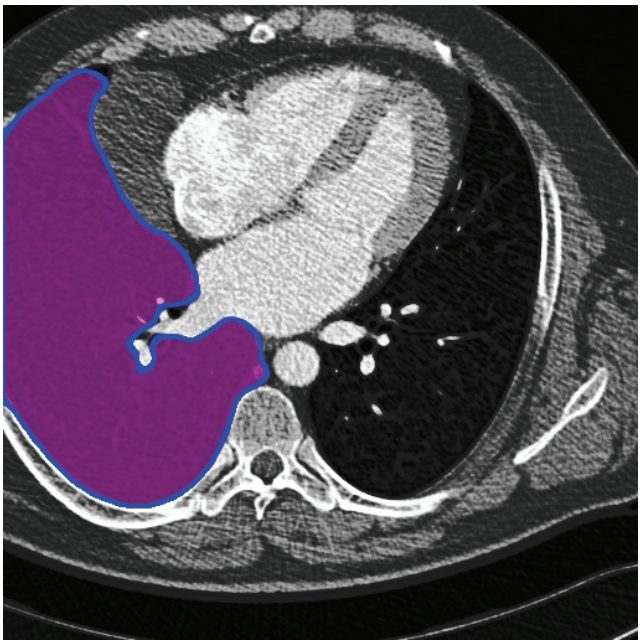


Рис. 7. Итоговая сегментация области интереса. Синим цветом выделена граница области, полученная с использованием сплайна, фиолетовым – итоговая маска области интереса.

Таблица 1. Точность сегментации левого желудочка для рассматриваемых алгоритмов

SPPS 20-10-5	**SPPS 16-8-4**	**SPPS 12-6-3**
0.93 ± 0.03	0.91 ± 0.07	0.89 ± 0.10
SPPS 8-4-2	**RG**	**AC**
0.86 ± 0.11	0.88 ± 0.09	0.71 ± 0.17

суперпикселя. Проверка модуля тангенса угла наклона на превышение некоторого порогового значения позволяет точно детектировать границы и быть устойчивым к шумам.

Проверка на наличие границы производится на ребрах и диагоналях квадрата. Если граница обнаружена на основании выполнения хотя бы одного из условий, то решение о принадлежности суперпикселя к региону отклоняется. Таким образом достигается повышение надежности метода.

Как показано на рисунке 4, каждый суперпиксель имеет шесть сегментов (*AB*, *BC*, *CD*, *AD*, *AC* и *BD*). Каждый отрезок формируется массивом пикселей. Таким образом, решение сводится к поиску точки пересечения с границей области интереса на отрезке. Такой подход значительно сокращает время выполнения алгоритма. Еще одним преимуществом предлагаемого решения является возможность применения любого набора методов для поиска пересечения. Стоит отметить, что, если хотя бы один сегмент пересек границу исследуемой области, алгоритм не обрабатывает остальные отрезки, что позволяет значительно увеличить производительность без потери в точности.

Суперпиксели расставляются на изображении до тех пор, пока существует хотя бы один суперпиксель, который можно присоединить к итоговому региону без пересечения границ области интереса. Если нет ни одного суперпикселя с заданным размером, которые можно добавить, размер блока уменьшается вдвое, и процесс распространения продолжается. Важно отметить, что суперпиксели меньшего размера расставляются только в свободные области.

Если блоки достигли минимального размера и больше нет ни одной группы пикселей, которую можно присоединить к региону без пересечения границы области интереса, то происходит формирование контура из суперпикселей. Суперпиксель принадлежит контуру только в том случае, если он имеет минимальный размер и одна из его сторон или диагоналей пересекает границу области интереса. Пример расстановки суперпикселей разных размеров и поиска внешних суперпикселей показан на рисунке 5. Зеленые точки указывают на пересечение суперпикселей с границей области интереса.

Процедура уменьшения размера суперпикселя может повторяться несколько раз, до тех пор, пока не будет достигнут минимальный размер суперпикселя. Минимальный размер суперпикселя является одним из параметров метода SPPS. Как можно заметить на рисунке 5, последним этапом алгоритма является объединение суперпикселей региона с наименьшим размером (фиолетовые блоки). Они расположены достаточно близко к границе сегментируемой области, но не пересекают ее.

3.5. Формирование контура региона

Как было описано выше, с целью формирования контура области интереса осуществляется обход внешних суперпикселей, которые не были прикреплены к региону. Набор внешних суперпикселей образует контур в соответствии с обязательным условием, описанным в пункте 3.4. При определении контура обход происходит по часовой стрелке. Точки пересечения суперпикселя и границы области интереса сохраняются, создавая массив узловых пограничных точек. Аналогичная процедура выполняется для обхода внутренних границ региона, если таковые существуют.

Имея список узловых точек, строится кубический сплайн Эрмита, который описывает внутренний и внешний контуры сегментируемой области (см. рисунок 6). Итоговая маска сегментируемой области показана фиолетовым цветом на рисунке 7.

Точкой пересечения является пиксель, который лежит на ребрах или диагонали суперпикселя и находится в точке пересечения с границей исследуемой области. Алгоритм определения пересечения с границей области интереса описан в пункте 3.4. После того, как все точки пересечения получены существует несколько вариантов формирования итогового контура. Наиболее примитивный – соединение каждой точки отрезками. Однако, такой подход значительно снижает итоговую точность сегментации. Оптимальный подход с точки зрения скорости и точности – построение регрессии полученных точек контура. Проведя исследование и анализ разных подходов, было принято решение отказаться от использования линейного сплайна, поскольку он дает значительную ошибку при построении границ контура. *B*-сплайн также снижает точность сегментации, так как не проходит через точки экстремума. Эмпирически было показано, что применение кубического эрмитова сплайна дает наиболее точный результат сегментации, что достигается за счет возможности прохода через узловые точки и подбора достаточной кривизны на выпуклых и вогнутых участках изображения.

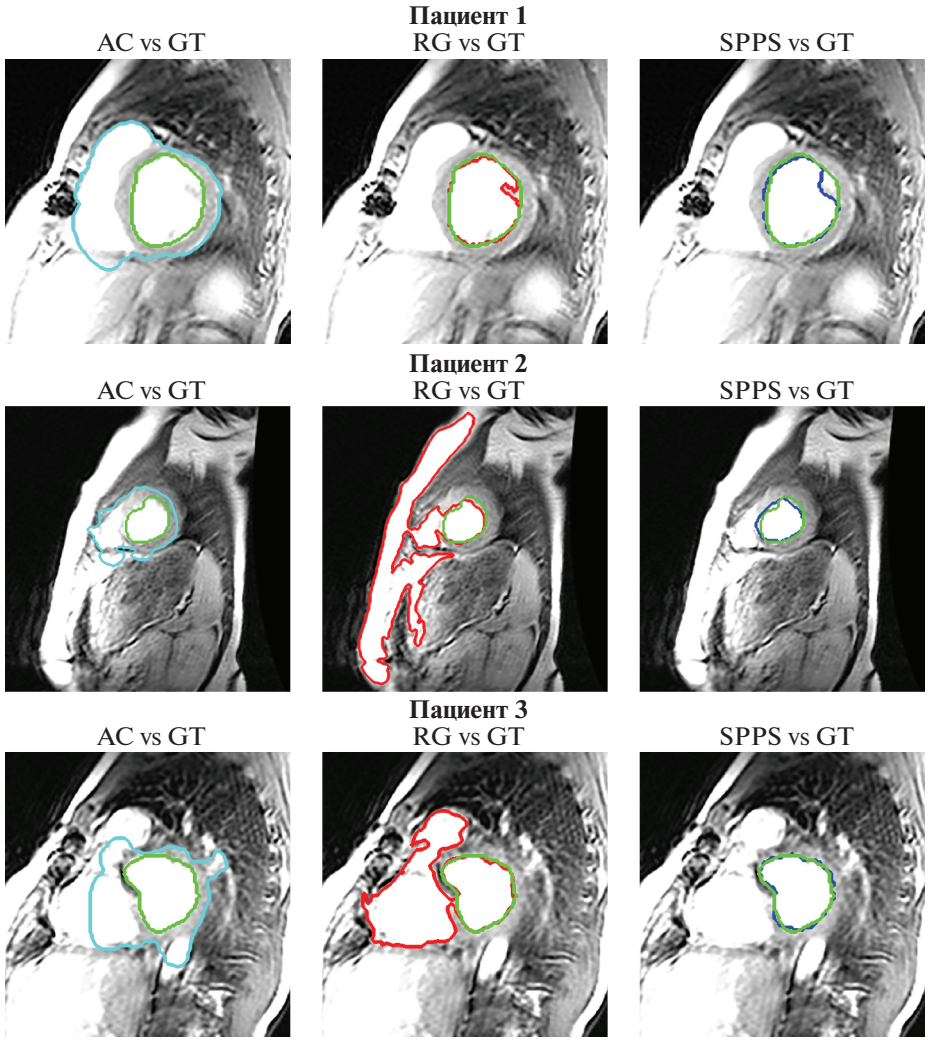


Рис. 8. Сегментация левого желудочка в сравнении с эталонной сегментацией (зеленый контур).

Таким образом, результатом метода сегментации является кубический сплайн, описывающий контур исследуемой области. Такое аналитическое представление может быть более предпочтительным, чем представление сегментированной области в форме маски.

3.6. Определение размеров суперпикселей

Методы сегментации на основании выращивания регионов обладают свойством “вытекания” через нечеткие и размытые границы. Некоторые модификации метода позволяют избежать этот эффект, однако, такие методы имеют высокую вычислительную сложность [24]. Предлагаемый метод SPPS позволяет избежать эффекта “вытекания” путем увеличения минимального размера суперпикселя. Однако, стоит отметить, что размер минимального блока влияет на сглаженность итогового контура. Таким образом, чем больше сторона суперпикселя, тем более сглаженный и менее детализированный контур формируется на выходе. Обратная ситуация происходит при уменьшении размера суперпикселя, а именно, возрастает детализация границы, однако вероятность появления нежелательного эффекта разрыва границы увеличивается.

Начальный размер суперпикселя оказывает большое влияние на скорость работы алгоритма и напрямую зависит как от сегментируемой области, так и от разрешения изображения. Начальный размер суперпикселя K в предложенном алгоритме был рассчитан на основании работы [22]. Формула его рассчета представлена ниже:

$$K = \frac{N}{SF}, \tag{3.4}$$

где N – число пикселей изображения, SF – размер наименьшего сегментируемого объекта.

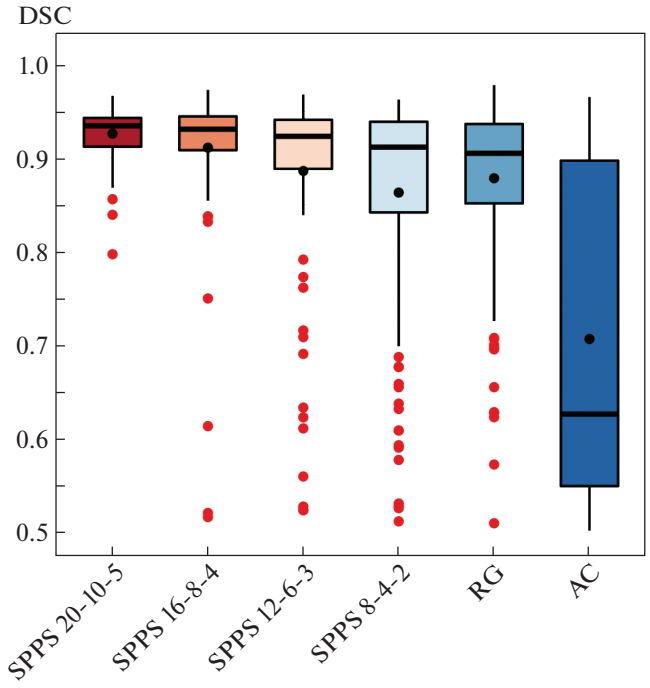


Рис. 9. Точность сегментации левого желудочка для рассматриваемых алгоритмов.

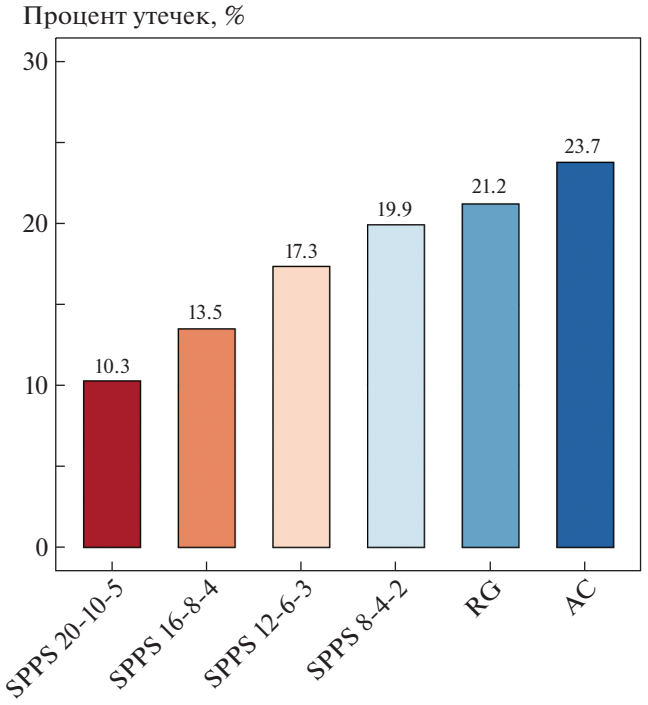


Рис. 10. Количество случаев ошибочной сегментации левого желудочка.

3.7. Сложность алгоритма

В стандартной реализации алгоритма выращивания регионов требуется по меньшей мере N^2 операций для того, чтобы обработать изображение размером $N \times N$ пикселей, что соответствует сложности алгоритма $O(n^2)$. В свою очередь, в предлагаемом алгоритме, для обработки изображения размером $N \times N$ пикселей, требуется от $3 \times N$ до $5 \times N$ операций, в зависимости от параметров алгоритма. Единственным исключением является первый суперпиксель, для которого необходимо обработать $6 \times N$ пикселей, то есть все диагонали и ребра суперпикселя. Таким образом, асимптотическая сложность алгоритма SPPS равна $O(n)$. Как будет показано ниже, прирост в скорости вычислений при подобных переходах особенно заметен на изображениях с высоким разрешением.

Таблица 2. Точность сегментации опухоли головного мозга для рассматриваемых алгоритмов

SPPS 20-10-5	**SPPS 16-8-4**	**SPPS 12-6-3**
0.89 ± 0.07	0.88 ± 0.08	0.88 ± 0.08
SPPS 8-4-2	**RG**	**AC**
0.87 ± 0.09	0.86 ± 0.10	0.83 ± 0.13

4. РЕЗУЛЬТАТЫ

В данном разделе приведены результаты оценки точности и скорости предлагаемого метода в зависимости от выбора его параметров. В качестве способа определения точности сегментации был использован коэффициент подобия Дайса (DSC, сокр. от Dice similarity coefficient).

4.1. Сегментация левого желудочка

Сегментация левого желудочка, выполненная предлагаемым методом SPPS, методом выращивания регионов RG, методом активных контуров AC и эталонной сегментации GT, представлена на рисунке 8. Как видно из рисунка 8, для пациентов 2 и 3 метод выращивания регионов сильно выходит за границы региона интереса, что не позволяет выполнить точную сегментацию. Аналогичная картина наблюдается для метода активных контуров. Несмотря на то, что подобные утечки могут быть нивелированы различного рода улучшениями, для работы с разрывами границ потребуются дополнительные вычислительные ресурсы. В свою очередь, метод SPPS позволяет избежать проблемы протекания в местах с низкой интенсивностью на границе за счет управления размером суперпикселей.

С целью оценки точности сегментации и времени обработки левого желудочка был использован набор данных, состоящий из 156 двухмерных

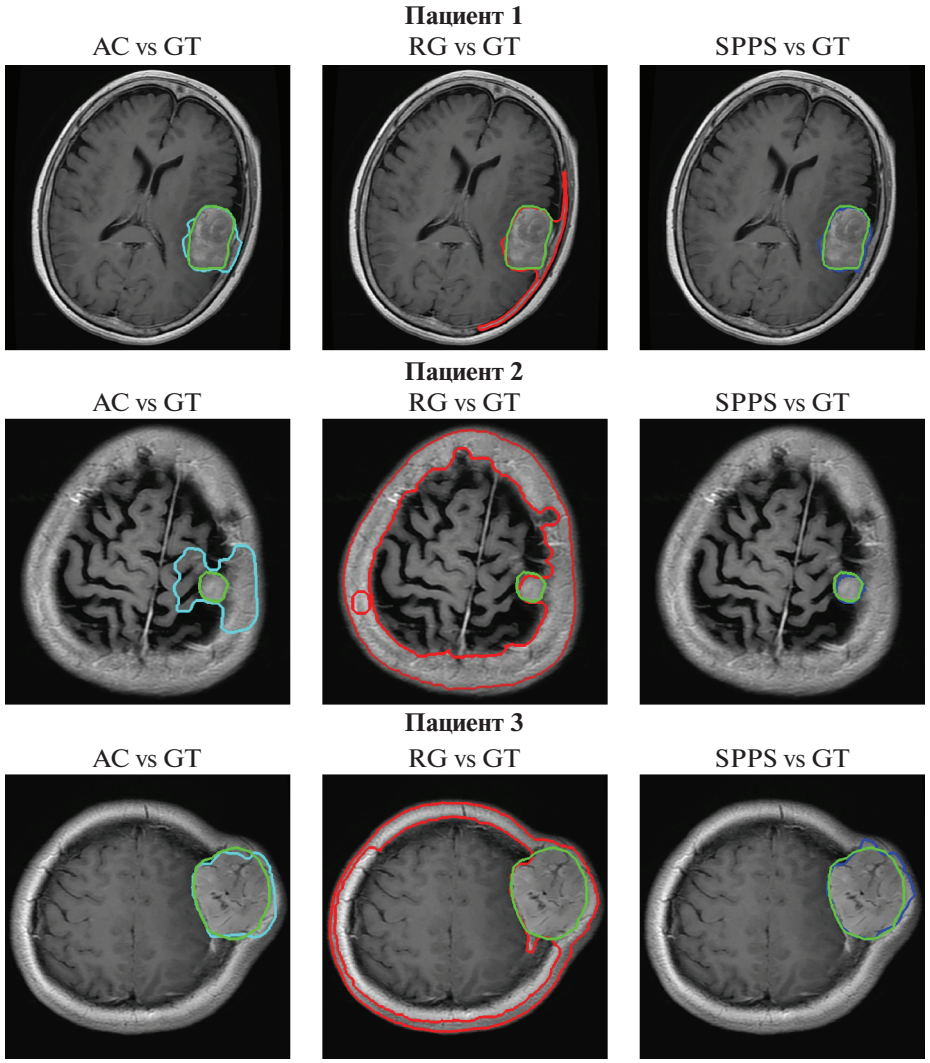


Рис. 11. Сегментация опухоли мозга в сравнении с эталонной сегментацией (зеленый контур).

изображений. Точность сегментации левого желудочка для метода SPPS с различными параметрами размеров суперпикселей показана на рисунке 9 и в таблице 1.

Кроме того, проведен анализ случаев "протекания" методов через границу области интереса. Такой анализ построен на основе оценки резкого снижения коэффициента Дайса (более 50%). Результаты данного анализа приведены на рисунке 10.

Как показано на рисунке 9 и в таблице 1, значения коэффициента Дайса для предлагаемого метода SPPS с размерами суперпикселей 8–4–2, 12–6–3 и метода выращивания регионов существенно не отличаются. Однако, интерквартильный размах значительно меньше у предлагаемого метода SPPS для параметров 20–10–5 и 16–8–4, чем у остальных методов. На исследуемом наборе данных метод активных контуров показал себя ненадежным, что подтверждается большим разма-

махом коэффициента Дайса. Следует отметить, что средняя точность метода SPPS выше в связи с тем, что подход к сегментации на основе суперпикселей позволяет избежать "протекания" через разрывы границ.

Более высокая эффективность предложенного метода косвенно подтверждается его надёжностью при сегментации регионов с разрывами. Последнее подтверждается низким уровнем утечек (см. рисунок 10). В 21% исследованных случаев, контур, полученный посредством метода выращивания регионов, выходит за пределы области интереса. Последнее обусловлено размытыми границами или разрывами, что подтверждает низкий уровень надежности данного метода. Схожая картина наблюдается для метода сегментации на основании активных контуров, где процент протекания составил почти 24%. В свою очередь, подбор

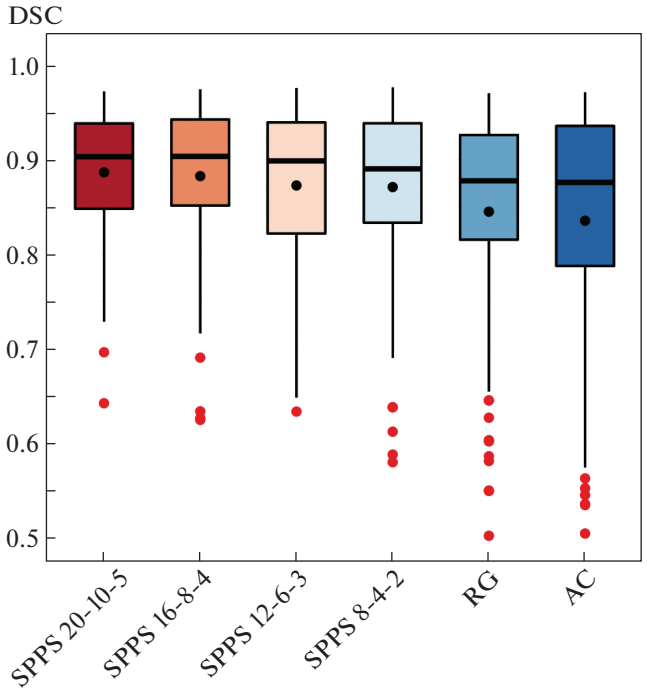


Рис. 12. Точность сегментации опухоли головного мозга для рассматриваемых алгоритмов.

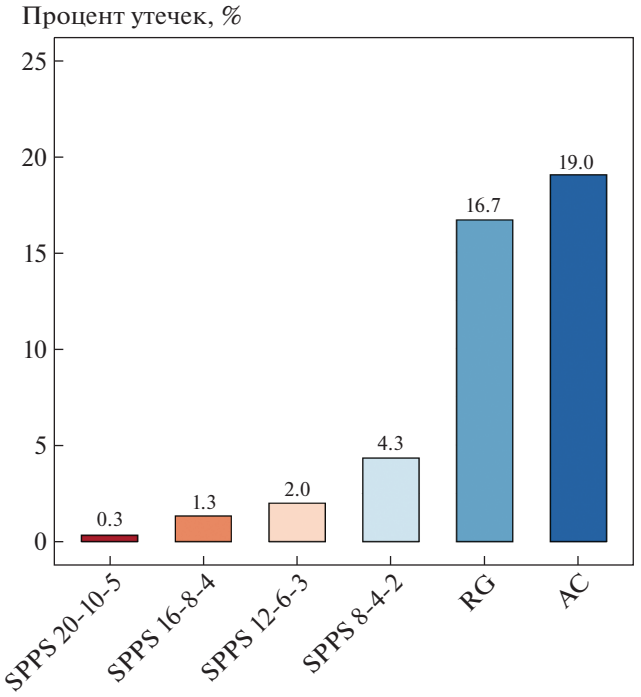


Рис. 13. Количество случаев ошибочной сегментации опухоли головного мозга.

оптимального размера суперпикселя в предлагаемом методе SPPS позволяет снизить протекание.

4.2. Сегментация опухоли мозга

Сегментация опухоли головного мозга, выполненная представленным методом SPPS, методом выращивания регионов RG, методом активных контуров AC и эталонной сегментации GT, представлены на рисунке 11. При исследованиях сегментации опухоли было выявлено, что алгоритм выращивания регионов RG сохраняет негативную тенденцию протекания через разрывы или размытия границ. Как показано на рисунке 11, костная ткань ошибочно сегментирована для трех представленных пациентов. С другой стороны, метод активных контуров AC выполняет сегментацию эффективнее на данных мозга, чем на данных сердца. В свою очередь, метод SPPS позволяет настраивать размер суперпикселей таким образом, что разрывы границ не оказывали негативного влияния на итоговую точность сегментации.

С целью оценки точности сегментации и расчета времени обработки опухоли головного мозга был использован набор данных, состоящий из 300 двухмерных изображений. Точность сегментации опухоли головного мозга для метода SPPS с различными параметрами размеров суперпикселей показана на рисунке 12 и в таблице 2. Кроме того, общее число случаев с низкой точностью, показано на рисунке 13.

Как видно из таблицы 2, существует корреляция между размерами суперпикселей и точностью. Чем меньше размер начального суперпикселя, тем ниже общая точность сегментации. Подобный эффект связан с тем, что в ряде изображений интенсивность пикселей опухоли не отличается от интенсивности костных тканей. Таким образом, при уменьшении размера суперпикселя увеличивается шанс попасть в область слияния региона интереса и заднего фона.

4.3. Время выполнения алгоритма

Для того, чтобы более наглядно продемонстрировать разницу во времени выполнения предлагаемого метода SPPS с методом выращивания регионов RG и методом активных контуров AC были созданы синтетические изображения различного размера. Синтетическое изображение состоит из белого круга в центре и заднего фона черного цвета. Очевидно, точность сегментации таких данных близка к 100%. Однако подобный подход позволяет продемонстрировать быстродействие методов. Как было указано выше, синтетическое изображение было сгенерировано в разных размерах. В результате тестирования методов получено время сегментации изображений разного размера. Данные результаты представлены на рисунке 14.

Для сравнения предлагаемого метода SPPS с методами RG и AC были использованы следующие

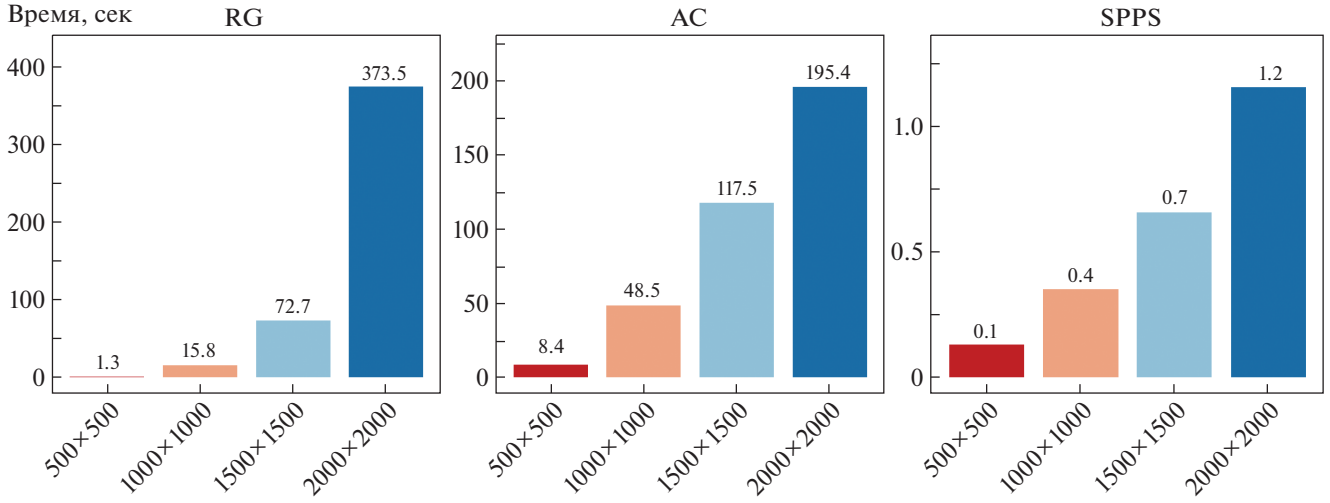


Рис. 14. Среднее время сегментации в зависимости от размера изображения.

параметры: размер начального суперпикселя – 16 пикселей, последующие размеры суперпикселей 8 и 4 пикселей. Для методов выращивания регионов и активных контуров выбраны настройки по умолчанию. Видно, что на относительно малых изображениях время сегментации методов RG и AC приемлемое. Однако для изображений с разрешением более 1000 × 1000 время сегментации этих методов становится сравнительно большим. При увеличении разрешения разрыв во времени выполнения только увеличивается. Таким образом, предлагаемый метод SPPS в силу своей низкой вычислительной сложности, показывает возможность эффективного решения задач сегментации изображений высокого разрешения.

5. ЗАКЛЮЧЕНИЕ

Предложенный в работе метод показывает высокую точность и производительность при сегментации изображений с высоким разрешением, медицинских данных с дефектами границы области интереса или изображений с низкой контрастностью. Кроме того, метод может применяться для разметки данных в полуавтоматическом режиме. Наиболее значимым параметром алгоритма, влияющим на быстродействие, является максимальный и минимальный размеры суперпикселей. Увеличение размера наибольшего суперпикселя ускоряет обработку изображения, но снижает детализацию контура. Другой важной особенностью предложенного метода является масштабируемость. Обуславливается это тем, что комбинации различных методов могут быть использованы для обнаружения пересечения с границей исследуемой области, что потенциально повысит надежность процесса сегментации. В качестве дополнительного алгоритма для обнаружения границ могут применяться методы машинного обучения, например, одномерные нейронные сети. Следует также отметить, что данный метод может быть модифицирован для выполнения трехмерной сегментации.

БЛАГОДАРНОСТИ

Данная работа выполнена в рамках государственного задания “Наука” № FFSWW-2020-0014 “Разработка научных основ технологии роботизированной мультипараметрической томографии на основе методов обработки больших данных и машинного обучения для исследования перспективных композиционных материалов”.

СПИСОК ЛИТЕРАТУРЫ

1. *Huang X., Tsechpenakis G.* Medical Image Segmentation // Adeanced Materials Research. 2009. № i. P. 1–35.
2. *Rogowska J.* Oeereiew and Fundamentals of Medical Image Segmentation // Handbook of Medical Imaging, Elseeier. 2000. P. 69–85.
3. *Pham D.L., Xu C., Prince J.L.* Current Methods in Medical Image Segmentation // Annual Reeiew of Biomedical Engineering. 2000. V. 2. № 1. P. 315–337.
4. *Kang H.C., Lee J., Shin J.* Automatic Four-Chamber Segmentation Using Leeel-Set Method and Split Energy Function // Healthcare Informatics Research. 2016. V. 22. № 4. P. 285.
5. *Soliman A. et al.* Accurate Lungs Segmentation on CT Chest Images by Adaptiee Appearance-Guided Shape Modeling // IEEE Transactions on Medical Imaging. 2017. V. 36. № 1. P. 263–276.
6. *Oktay O. et al.* Anatomically Constrained Neural Networks (ACNNs): Application to Cardiac Image Enhancement and Segmentation // IEEE Transactions on Medical Imaging. 2018. V. 37. № 2. P. 384–395.

7. *Pereira S. et al.* Brain Tumor Segmentation Using Coneolutional Neural Networks in MRI Images // IEEE Transactions on Medical Imaging. 2016. V. 35. № 5. P. 1240–1251.
8. *Kamnitsas K. et al.* Efficient Multi-Scale 3D CNN with Fully Connected CRF for Accurate Brain Lesion Segmentation // Medical Image Analysis. 2016. V. 36. P. 61–78.
9. *Litjens G. et al.* A Sureey on Deep Learning in Medical Image Analysis // Medical Image Analysis. 2017. V. 42. P. 60–88.
10. *Daniloe E.E. et al.* Catheter detection and segmentation in eolumetric ultrasound using SEM and GLCM // IEEE SciEis Contest. 2018. V. 10. № 4. P. 30–39.
11. *Daniloe E.E. et al.* Segmentation Algorithm Based on Square Blocks Propagation // Proceedings of the 29th International Conference on Computer Graphics and Eision / ed. Galaktionoe E. et al. Aachen: CEUR Workshop Proceedings. 2019. P. 148–154.
12. *Ballard D.H., Brown C.M.* Computer Eision, Prentice Hall, 1982. 523 p.
13. *Adams R., Bischof L.* Seeded Region Growing // IEEE Transactions on Pattern Analysis and Machine Intelligence. 1994. V. 16. № 6. P. 641–647.
14. *Tang J.* A color image segmentation algorithm based on region growing // 2nd International Conference on Computer Engineering and Technology, IEEE. 2010. V. 6. P. 634–637.
15. *Park J.G., Lee C.* Skull stripping based on region growing for magnetic resonance brain images // Neuroimage. 2009. V. 47. № 4. P. 1394–1407.
16. *Chieerton J. et al.* Statistical morphological skull stripping of adult and infant MRI data // Computers in Biology and Medicine. 2007. V. 37. № 3. P. 342–357.
17. *Roy S., Maji P.* A simple skull stripping algorithm for brain MRI // 2015 Eighth International Conference on Adeances in Pattern Recognition (ICAPR). IEEE. 2015. P. 1–6.
18. *Isa N.A.M. et al.* Automatic Detection of Breast Tumours from Ultrasound Images Using the Modified Seed Based Region Growing Technique // Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics). 2005. V. 3682 LNAI. P. 138–144.
19. *Dehmeshki J. et al.* Segmentation of Pulmonary Nodules in Thoracic CT Scans: A Region Growing Approach // IEEE Transactions on Medical Imaging. 2008. V. 27. № 4. P. 467–480.
20. *Crommelinck S. et al.* SLIC superpixels for object delineation UAE data // ISPRS Annals of the Photogrammetry, Remote Sensing and Spatial Information Sciences. 2017. V. 4. № 2W3. P. 9–16.
21. *Csillik O.* Fast Segmentation and Classification of Eery High Resolution Remote Sensing Data Using SLIC Superpixels // Remote Sensing. 2017. V. 9. № 3. P. 243.
22. *Duong T.H., Hoberock L.L.* DUHO image segmentation based on unseeded region growing on superpixels // 2018 IEEE 8th Annual Computing and Communication Workshop and Conference (CCWC). IEEE. 2018. P. 558–563.
23. *Saxen F., Al-Hamadi A.* Superpixels for Skin Segmentation // Workshop Farbbildeerarbeitung. 2014. P. 153–159.
24. *Ren X., Malik J.* Learning a classification model for segmentation // Proceedings of the Ninth IEEE International Conference on Computer Eision. 2003. V. 2. P. 10.
25. *Achanta R. et al.* SLIC Superpixels Compared to State-of-the-Art Superpixel Methods // IEEE Transactions on Pattern Analysis and Machine Intelligence. 2012. V. 34. № 11. P. 2274–2282.
26. *Andreopoulos A., Tsotsos J.K.* Efficient and generalizable statistical models of shape and appearance for analysis of cardiac MRI // Medical Image Analysis. 2008. V. 12. № 3. P. 335–357.
27. *Cheng J. et al.* Enhanced Performance of Brain Tumor Classification eia Tumor Region Augmentation and Partition // Public Library of Science One / ed. Zhang D. 2015. V. 10. № 10.
28. *Cheng J. et al.* Retrieeal of Brain Tumors by Adaptiee Spatial Pooling and Fisher Eector Representation // Public Library of Science One. 2016. V. 11. № 6.
29. *Wang L. et al.* Left Eentricle: Fully Automated Segmentation Based on Spatiotemporal Continuity and Myocardium Information in Cine Cardiac Magnetic Resonance Imaging (LE-FAST) // BioMed Research International. 2015. V. 2015. P. 1–9.

ПРОГРАММИРОВАНИЕ, 2020, № 3, с. 16–23

КОМПЬЮТЕРНАЯ ГРАФИКА И ВИЗУАЛИЗАЦИЯ

УДК 004.92

ПАРАЛЛЕЛЬНЫЕ РЕШЕНИЯ ПАРАМЕТРИЧЕСКИХ ЗАДАЧ ГАЗОВОЙ ДИНАМИКИ С ПОМОЩЬЮ ТЕХНОЛОГИИ DVM/DVMH

© 2020 г. А. Е. Бондарев[a],*, В. А. Галактионов[a],, А. Е. Кувшинников[a],*****

[a] *Институт прикладной математики им. М.В. Келдыша РАН*
125047 Москва, Миусская пл., 4, Россия
**E-mail: bond@keldysh.ru*
***E-mail: vlgal@gin.keldysh.ru*
****E-mail: kuvsh90@yandex.ru*

Поступила в редакцию 18.12.2019 г.
После доработки 13.01.2020 г.
Принята к публикации 20.01.2020 г.

Работа представляет исследование эффективности применения технологии DVM/DVMH для организации параллельного решения параметрических задач газовой динамики. Решение параметрических задач базируется на использовании подхода построения обобщенного вычислительного эксперимента, предполагающего многократное решение задачи при вариации определяющих параметров. Такой подход эффективен только при использовании параллельных технологий, позволяющих организовать одновременное решение рассматриваемой задачи с различными входными данными в многозадачном режиме. Приведено описание проведенных численных экспериментов по решению тестовых одномерных и двумерных параметрических задач газовой динамики при вариации задаваемого числа вычислительных узлов, параметров этих узлов и выделяемых графических сопроцессоров. Представлены результаты расчетов для ряда тестовых задач в виде характеристик эффективности и ускорения.

DOI: 10.31857/S0132347420030036

1. ВВЕДЕНИЕ

В настоящее время развитие параллельных технологий является одним из наиболее актуальных направлений исследований. Идет активная работа по созданию экзафлопсного суперкомпьютера (10^{18} операций над числами с плавающей точкой в секунду). Для создания подобного суперкомпьютера необходимы новые энергоэффективные технологии [1], в число которых входит использование графических ускорителей и сопроцессоров.

Постоянно появляющиеся новые технологии требуют изменения программного кода, для более эффективного использования вычислительной мощности. Долговременная поддержка программ с внедрением новых технологий крайне затратная, поэтому появляются способы автоматического (или полуавтоматического) распараллеливания. Развитие технологий, позволяющих автоматическое (или полуавтоматическое) распараллеливание является крайне важным с точки зрения использования в параллельном режиме уже существующих программ, изначально созданных не для работы в режиме параллельных вычислений.

Разработанная в Институте прикладной математики им. М.В. Келдыша РАН высокоуровневая модель параллельного программирования DVMH (DVM for Heterogeneous systems) [2] предназначена для разработки параллельных программ численного моделирования на языках C-DVMH и Fortran-DVMH. Эти языки используют единую модель параллельного программирования (DVMH-модель) и являются расширением стандартных языков Си и Фортран спецификациями параллелизма, оформленными в виде директив компилятору. Поскольку директивы невидимы для стандартных компиляторов, программист может иметь одну программу и для последовательного, и для параллельного выполнения на ЭВМ разной архитектуры. Таким образом, единожды вставив в код последовательной программы DVMH-директивы в виде специальных комментариев, можно получить вариант той же программы для использования в режиме параллельных вычислений.

DVMH-модель [2–8] позволяет создавать эффективные параллельные программы (DVMH-программы) для гетерогенных вычислительных кластеров, в узлах которых в качестве вычислительных устройств наряду с универсальными многоядерными процессорами могут использо-

ваться ускорители (графические процессоры или сопроцессоры Intel Xeon Phi). При этом отображенные на узел вычисления могут автоматически распределяться между вычислительными устройствами узла с учетом их производительности. Компиляторы языков C-DVMH и Fortran-DVMH преобразуют входную программу в параллельную программу, использующую стандартные технологии программирования MPI, OpenMP и CUDA. В состав DVM-системы входят средства функциональной отладки и отладки эффективности DVMH-программ. Наличие такого эффективного инструмента для распараллеливания программ позволяет решать широкий спектр практических задач.

К масштабным задачам такого рода следует отнести построение обобщенного вычислительного эксперимента. Теоретические основы, методы и алгоритмы построения обобщенного вычислительного эксперимента подробно описаны в работах [9–11]. Обобщенный вычислительный эксперимент строится на основе проведения параметрических исследований и решения задач оптимизационного анализа. Параметрические численные исследования позволяют получать решение не для одной конкретной задачи математического моделирования, а для класса задач, заданного в многомерном пространстве определяющих параметров. Оптимизационный анализ основан на массовом решении обратных задач при изменяющихся в определенных диапазонах определяющих параметрах рассматриваемого класса задач. Результаты подобных вычислений представляют собой многомерные массивы, размерность которых соответствует количеству определяющих параметров. Для обработки и анализа этих многомерных массивов используются инструменты визуальной аналитики. Применительно к задачам вычислительной газовой динамики применение подобного подхода позволяет проведение анализа процессов возникновения, трансформации и распада нестационарных пространственно-временных структур (ударные волны, отрывные зоны) в потоках и исследование условий возникновения колебательных режимов. Также для задач вычислительной газовой динамики обработка и анализ дискретных численных решений во многих случаях позволяет построение приближенных зависимостей для ценных функционалов от определяющих параметров задачи.

Одним из важнейших и по сей день неоцененных в достаточной степени преимуществ параллельных вычислений является то, что они позволяют параллельно решать в многозадачном режиме одну и ту же задачу с разными входными данными. Подобное преимущество делает параллельные вычисления важнейшим инструментом для решения оптимизационных задач, обратных задач в оптимизационной постановке и задач параметрического поиска. Эти задачи являются основой построения обобщенного вычислительного эксперимента. Характерной общей чертой этих типов задач является то, что в вычислительном смысле они сводятся к массовому решению однотипных задач с изменяющимися входными параметрами. Это обстоятельство делает их удобными объектами для параллельных вычислений на основе многозадачного параллелизма. Без применения параллельных вычислений решение задач подобного типа не представляется возможным. Особенности задач подобного типа и практические примеры применения параллельных вычислений к данным типам задач подробно описываются в работах [12–14].

Продолжая исследования работ [12–14], данная работа имеет своей целью описание результатов проведенных вычислительных экспериментов параллельного решения параметрических задач газовой динамики с использованием технологии DVMH на гибридном вычислительном кластере.

Для исследований эффективности применения системы DVM/DVMH был выбран ряд тестовых задач. Все эти задачи рассматривались как параметрические, то есть, для определяющих (ключевых) параметров задачи в заданных диапазонах задавалось разбиение, и затем задачи решались в параллельном многозадачном режиме с помощью системы DVMH. Таким образом с помощью технологии DVM/DVMH для выбранных тестовых задач реализовывалось построение обобщенного вычислительного эксперимента.

Следует отметить, что для анализа эффективности применения технологии DVM/DVMH также использовался подход построения обобщенного вычислительного эксперимента, где в качестве ценных функционалов рассматривались характеристики эффективности и ускорения при вариации задаваемого числа вычислительных узлов, параметров этих узлов и выделяемых графических сопроцессоров, как определяющих параметров задачи.

Все выбранные задачи являются простыми и достаточно известными одномерными или двумерными задачами вычислительной газовой динамики. Рассмотрим эти задачи вместе с алгоритмами решения и полученные результаты.

2. ЛИНЕЙНОЕ УРАВНЕНИЕ БЮРГЕРСА

Рассмотрим краевую задачу, основанную на применении линейного уравнения Бюргерса, представленного в следующем виде:

$$\frac{\partial u}{\partial t} + \frac{\partial u}{\partial x} = \varepsilon \frac{\partial^2 u}{\partial x^2} + f(x,t)$$

Начальные и граничные условия представляются в виде:

$$u(x,t_0) = y(x,t_0),$$

$$u(0,t) = y(0,t), \quad u(L,t) = y(L,t),$$

где $y(x, t)$ – точное решение, описывающее функцию типа бегущей волны.

Точное решение $y(x, t)$ имеет вид:

$$y(x,t) = y(\xi) = e^{-d(1-2\xi)^2}, \quad \text{где} \quad \xi = (t-x)/t_1$$

Величины t_1 и d характеризуют ширину и крутизну волны и являются параметрами задачи. Функция $f(x, t)$ в правой части является выражением вида:

$$f(x,t) = -\frac{\varepsilon}{t_1^2}[16d^2(1-2\xi)^2 - 8d]e^{-d(1-2\xi)^2}.$$

Для решения данной задачи использовалась неявная конечно-разностная схема, подробно описанная в [15, 16]. При решении параметрической задачи параметры ε (коэффициент вязкости) и весовой коэффициент гибридной разностной схемы S_k [16] разбивались в определенных диапазонах, и для каждой пары значений решалась описанная выше задача. При проведении экспериментов по распараллеливанию данного программного кода с помощью DVM варьировались следующие параметры:

N – число MPI-процессов,

PPN – число MPI-процессов на один вычислительный узел.

В проведенных экспериментах число MPI-процессов N варьировалось от 1 до 32, а число MPI-процессов на один вычислительный узел PPN варьировалось от 1 до 8. Результаты представлены ниже в таблице 1.

Здесь и далее в таблицах будем использовать следующие обозначения:

N – число MPI-процессов,

PPN – число MPI-процессов, запускаемых на одном узле,

THR – число нитей, используемых каждым MPI-процессом,

CUDA – число графических ускорителей, используемых каждым MPI-процессом,

T – время в секундах, S – ускорение T_{serial}/T (или T_1/T),

E – эффективность параллелизации, определяемая как S/(N*THR) (S/N при THR=0).

Полученные результаты расчетов показали эффективность реализованного решения по многозадачному распараллеливанию. Одновременно в работе [15] отмечалось, что применяемая в программном коде решения уравнения Бюргерса неявная конечно-разностная схема (WW-схема) неудобна для распараллеливания с помощью подключения графических сопроцессоров. Для проведения следующей серии численных тестов ее было необходимо заменить на явный аналог с сохранением свойств аппроксимации и устойчивости. Для

Таблица 1. Уравнение Бюргерса – неявная схема

N	PPN	T	S	
serial	–	51.4	1	1
1	1	51.4	1.001	1.001
2	1	25.7	1.997	0.998
2	2	25.7	1.998	0.999
4	1	12.9	3.988	0.997
4	2	12.9	3.987	0.997
4	4	12.9	3.991	0.998
8	1	7.61	6.753	0.844
8	2	6.71	7.662	0.958
8	4	6.73	7.641	0.955
8	8	6.98	7.363	0.920
16	2	4.09	12.580	0.786
16	4	3.64	14.129	0.883
16	8	3.76	13.652	0.853
32	2	2.08	24.684	0.769
32	4	2.09	24.613	0.769
32	8	2.16	23.782	0.743

проведения дальнейших экспериментов по подключению графических сопроцессоров была проведена замена используемой в алгоритме конечно-разностной схемы. Неявная конечно-разностная схема [16] (WW-схема) была заменена в алгоритме и программном коде на явную схему типа Лакса–Вендроффа, имеющую второй порядок по пространству и времени:

$$u_i^{n+1} = u_i^n - \frac{\tau}{2h}(u_{i+1}^n - u_{i-1}^n) + \frac{\tau^2}{2h^2}(u_{i+1}^n - 2u_i^n + u_{i-1}^n) + \\ + \varepsilon\frac{\tau}{h^2}(u_{i+1}^n - 2u_i^n + u_{i-1}^n) + \tau f(x,t)$$

Для подобной конечно-разностной схемы был реализован программный код с применением директив как DVM, так и DVMH с подключением графических сопроцессоров.

Проведены исследования с увеличением числа узлов разбиения по пространству. Исследовалось влияние на эффективность и ускорение изменения параметров N, PPN, THR, CUDA. В таблице 2 приведены результаты численных экспериментов при следующих значениях параметров: узлов по пространству – 1501, шагов по времени– 7000, шагов по вязкости– 10.

Аналогичные результаты приведены в таблице 3 при следующих значениях параметров: узлов по пространству – 15001, шагов по времени– 7000, шагов по вязкости– 10.

Продолжая увеличивать число узлов сеточного разбиения по пространственной переменной, получаем результаты, представленные в таблице 4

Таблица 2. Уравнение Бюргерса – явная схема, 1501 узел

N	PPN	THR	CUDA	T	S	E
1	8	0	0	8.813	1	1
2	8	0	0	8.189	1.0760	0.5381
4	8	0	0	7.588	1.1614	0.2904
8	8	0	0	7.182	1.2271	0.1534
16	8	0	0	7.35	1.1991	0.0749
1	1	0	1	12.372	0.7123	
1	1	2	0	7.467	1.1803	0.5901
1	2	4	0	6.459	1.3645	0.2951
2	2	4	0	6.063	1.4536	0.1475

Таблица 3. Уравнение Бюргерса – явная схема, 15001 узел

N	PPN	THR	CUDA	T	S	E
1	8	0	0	40.539	1	1
2	8	0	0	24.327	1.6664	0.8332
4	8	0	0	15.915	2.5472	0.6368
8	8	0	0	11.255	3.6019	0.4502
16	8	0	0	9.347	4.3371	0.2711
32	8	0	0	8.324	4.8701	0.1522
1	1	2	0	23.549	1.7215	0.8607
1	2	4	0	14.45	2.8055	0.7014
2	2	4	0	10.058	4.0305	0.5038
4	2	4	0	9.478	4.2772	0.2673
8	2	4	0	8.41	4.8203	0.1506
1	1	0	1	12.799	3.1674	
2	1	0	1	18.251	2.2212	

для параметров: узлов по пространству – 150001, шагов по времени – 7000, шагов по вязкости – 10.

По результатам можно сделать вывод о том, что на грубой сетке использование графических сопроцессоров не только не дает никакого эффекта, но даже замедляет выполнение тестового расчета. При резком увеличении количества узлов расчетной сетки обеспечивается эффективная работа параллельных узлов, при этом подключение графических сопроцессоров также позволяет резко повысить скорость расчетов.

На рисунках 1 и 2 представлены сводные результаты расчетных тестов для решения линейного уравнения Бюргерса. На рисунке 1 представлены графики зависимости эффективности от увеличения числа узлов вычислительного кластера для различного количества узлов расчетной сетки. На рисунке 2 представлены графики зависимости ускорения от увеличения числа узлов вычислительного кластера для различного количества узлов расчетной сетки. Расчеты ускорения представлены для трех вариантов задания количества узлов расчетной сетки.

3. КВАЗИЛИНЕЙНОЕ УРАВНЕНИЕ БЮРГЕРСА

По аналогичной схеме были проведены тестовые расчеты для квазилинейного уравнения Бюргерса:

$$\frac{\partial u}{\partial t} + \frac{\partial (u^2/2)}{\partial x} = \varepsilon \frac{\partial^2 u}{\partial x^2}$$

Решением здесь является сглаженная ударная волна:

$$u = \frac{a + b\exp\left(\dfrac{a-b}{2\varepsilon}(x - x_0 - Dt)\right)}{1 + \exp\left(\dfrac{a-b}{2\varepsilon}(x - x_0 - Dt)\right)},$$

где $D = (a + b)/2$.

Задача решалась при следующем выборе параметров: $a = 1.0$, $b = 0.2$, $\varepsilon = 0.05$, $x_0 = 5.0$.

В качестве сеточного разбиения задавалось: узлов по пространству – 801, шагов по времени – 9000. Аналогично предыдущей задаче проводилось увеличение сеточного разбиения по пространству до

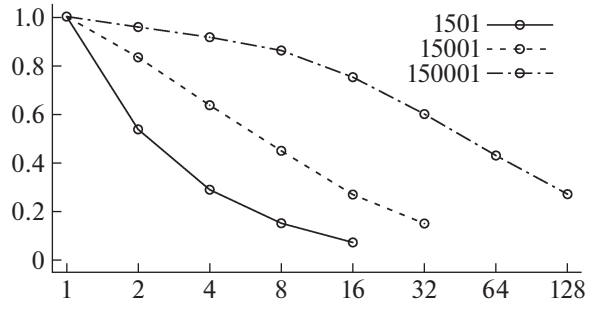


Рис. 1. Значение эффективности при увеличении числа узлов для различного количества узлов расчетной сетки.

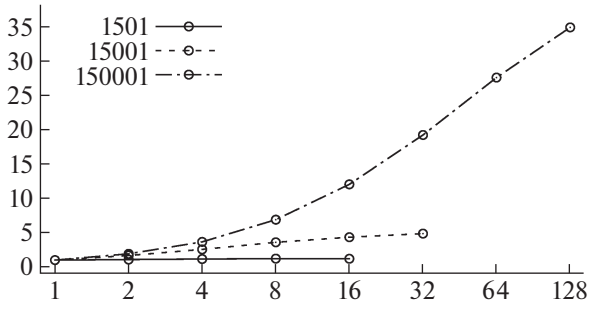


Рис. 2. Значение ускорения при увеличении числа узлов для различного количества узлов расчетной сетки.

1000001 узлов. Результаты представлены в таблицах 5 и 6.

Из полученных результатов можно сделать общий вывод. Использование подключения графических сопроцессоров дает резкий эффект ускорения. Но это происходит только в случае достаточно большого количества расчетных узлов. Для грубой сетки, то есть для малого количества расчетных узлов сетки, распараллеливание на несколько процессоров, а тем более подключение графических сопроцессоров приводит к замедлению расчетов по сравнению с последовательным вариантом. Таким образом, расчетная сетка должна быть настолько подробной, чтобы обеспечивать загрузку графических сопроцессоров.

4. ОДНОМЕРНЫЕ УРАВНЕНИЯ ГАЗОВОЙ ДИНАМИКИ

Решалась начально-краевая задача для уравнений динамики невязкого газа (уравнения Эйлера). Исходные уравнения для одномерного течения записывались в консервативной форме:

$$\frac{\partial \rho}{\partial t} + \frac{\partial \rho v}{\partial x} = 0,$$

$$\frac{\partial \rho v}{\partial t} + \frac{\partial (\rho v^2 + p)}{\partial x} = 0,$$

$$\frac{\partial \rho E}{\partial t} + \frac{\partial ((\rho E + p)u)}{\partial x} = 0.$$

Здесь $\rho E = \rho u^2/2 + \rho\varepsilon$ – плотность полной энергии газа, ε – внутренняя энергия газа, $p = \rho\varepsilon(\gamma - 1)$ – давление газа, γ – показатель адиабаты газа.

Вектор начальных условий задавался следующим образом:

$$(\rho, u, p)^T = \begin{cases} (1.0, 0.75, 0.1)^T, & x \le 2.0, \\ (0.125, 0.0, 0.1)^T, & x > 2.0 \end{cases}$$

В подобной постановке данная задача является задачей о распаде произвольного разрыва (задачей Сода) [17, 18], для которой есть точное решение.

Рассматривается отрезок [0,8], время расчета $t = 2.4$, шаг по пространству $h = 0.001$, шаг по времени $\tau = 0.0002$.

Для решения данной системы в используемом программном коде применяется схема типа WENO (взвешенные, существенно не осциллирующие схемы), имеющая 5 порядок аппроксимации по пространственной переменной и 3-й порядок аппроксимации по времени [19, 20].

Результаты численных экспериментов представлены в таблице 7.

Таблица 4. Уравнение Бюргерса – явная схема, 150001 узел

N	PPN	THR	CUDA	T	S	E
1	8	0	0	352.89	1	1
2	8	0	0	184.334	1.9144	0.9572
4	8	0	0	96.38	3.6614	0.9154
8	8	0	0	51.233	6.8879	0.8610
16	8	0	0	29.35	12.0235	0.7515
32	8	0	0	18.354	19.2269	0.6008
64	8	0	0	12.799	27.5717	0.4308
128	8	0	0	10.119	34.8740	0.2725
1	2	4	0	94.756	3.7242	0.9311
2	2	4	0	50.522	6.9849	0.8731
4	2	4	0	29.53	11.9502	0.7469
8	2	4	0	18.418	19.1601	0.5988
16	2	4	0	12.959	27.2313	0.4255
32	2	4	0	10.503	33.5990	0.2624
1	1	0	1	19.53	18.0691	
2	1	0	1	21.226	16.6254	

Таблица 5. Квазилинейное уравнение Бюргерса – явная схема, 801 узел

N	THR	CUDA	T	S	E
serial			0.5354	1	1
1	0	0	0.6344	0.844	0.844
2	0	0	1.0371	0.5163	0.2581
4	0	0	2.3362	0.2292	0.0573
1	3	0	1.2987	0.4123	0.1374
1	6	0	1.2801	0.4183	0.0697
1	0	1	6.9457	0.0771	
1	0	2	7.2463	0.0739	

На рисунке 3 представлена величина ускорения для данной задачи при вариации числа процессов и подключении CUDA.

Видно, что при увеличении числа процессов ускорение также увеличивается. Подключение одного графического процессора по ускорению равно параллельной работе 8 процессов. Подключение второго графического процессора не дает эффекта.

5. УРАВНЕНИЯ НАВЬЕ–СТОКСА

После проведения тестовых расчетов для одномерных задач было решено провести аналогичное исследование эффективности для двумерной задачи.

Таблица 6. Квазилинейное уравнение Бюргерса – явная схема, 100001 узел

N	PPN	THR	CUDA	T	S	E
1	1	0	0	36.7523	1	1
2	1	0	0	16.7374	2.1958	1.0979
2	2	0	0	17.2453	2.1312	1.0656
4	2	0	0	9.2772	3.9616	0.9904
4	4	0	0	9.8062	3.7479	0.937
8	4	0	0	5.8577	6.2742	0.7843
8	8	0	0	8.2898	4.4334	0.554
1	1	2	0	18.1995	2.0194	1.0097
1	1	4	0	10.6017	3.4666	0.8667
1	1	6	0	8.8302	4.1621	0.6937
1	1	8	0	8.1091	4.5322	0.5665
1	1	0	1	9.8264	3.7402	
1	1	0	2	9.6775	3.7977	

Таблица 7. Задача о распаде произвольного разрыва

N	PPN	THR	CUDA	T	S	E
serial				246.14	1	1
1	12	0	0	259.46	0.95	0.95
2	12	0	0	133.65	1.84	0.92
4	12	0	0	69.79	3.53	0.88
8	12	0	0	38.82	6.34	0.79
12	12	0	0	27.58	8.93	0.74
24	12	0	0	17.6	13.99	0.58
1	1	4	0	69.34	3.55	0.89
1	1	8	0	38.18	6.45	0.81
2	2	6	0	27.58	8.93	0.74
1	1	0	1	25.75	9.56	
2	2	0	1	29.84	8.25	

Таблица 8. Уравнения Навье–Стокса

N	PPN	THR	CUDA	T	S	E
1	12	0	0	6299.9	1	1
2	12	0	0	3171.29	1.987	0.993
4	12	0	0	1645.51	3.829	0.957
8	12	0	0	868.25	7.256	0.907
12	12	0	0	586.96	10.733	0.894
24	12	0	0	294.45	21.396	0.892
48	12	0	0	168.48	37.392	0.779

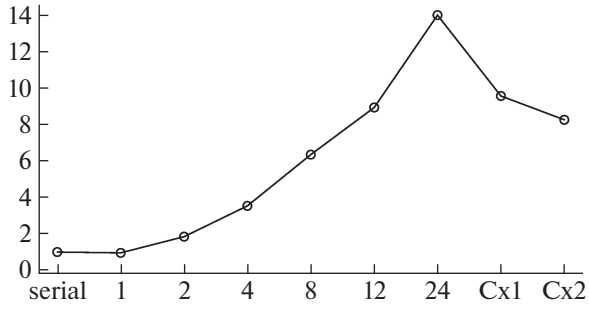


Рис. 3. Зависимость ускорения для задачи Сода от числа процессов подключения CUDA.

Рассматривалась задача вязкого течения в канале с заданием следующих граничных условий для решения полной системы уравнений Навье–Стокса для сжимаемого течения. На верхней границе задавалось условие жесткой стенки, на нижней границе – условие оси симметрии, на задней границе – условие экстраполяции, на входной границе – профиль скорости ламинарного пограничного слоя. В качестве начальных данных брались условия входной границы. Решалась система уравнений Навье–Стокса, варьировались три определяющих параметра задачи:

1) число Маха (1.5, 2.0, 2.5, 3.0);

2) число Рейнольдса (1000, 3162.278, 10000);

3) число Прандтля (0.72, 0.8, 0.9, 1.0).

Так как для решения данной параметрической задачи использовалась неявная схема [17], при решении рассматривалось только увеличение числа процессов аналогично линейному уравнению Бюргерса, описанному в разделе 2. Результаты представлены в таблице 7.

На рисунке 4 представлена зависимость ускорения для данной задачи от числа процессов N.

Как и в предыдущем случае, при увеличении количества параллельных процессов, увеличивается ускорение задачи. Однако, в отличие от предыдущего случая, данная задача является более ресурсозависимой, и позволяет загрузить 24 процессора. Эффективность параллельной работы 24 процессов составляет почти 90%.

6. ЗАКЛЮЧЕНИЕ

По итогам проведенных экспериментов следует заметить, что система DVM является эффективным инструментом, позволяющим полноценно использовать функционал организации параллельных вычислений с минимальными трудозатратами со стороны пользователя. Стоит заметить, что система вынуждена переводить фортрановские коды на C++, что накладывает ограничения на структуру программы, такие как обязательное

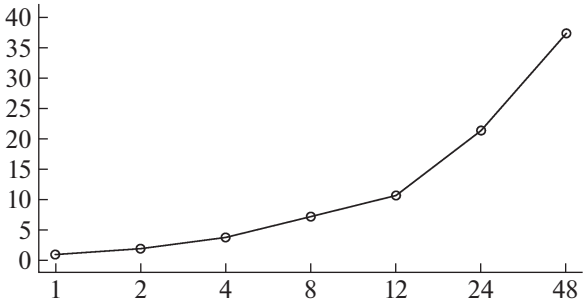


Рис. 4. Зависимость ускорения для уравнений Навье–Стокса от числа процессов.

использование pure функций и невозможность использования массивов без явного обозначения индексов. Если отбросить все эти проблемы и учесть, что приведенные выше задачи являются не настолько вычислительно сложными, как реальные 3D задачи, можно сказать, что DVM-система показывает высокую эффективность для решения параметрических задач газовой динамики. Это обстоятельство делает технологию DVM/DVMH в целом исключительно ценной для построения обобщенного вычислительного эксперимента.

По результатам проведенных расчетов можно сделать следующий вывод. При подключении графических ускорителей с помощью технологии DVMH возрастают накладные расходы на передачу информации. Поэтому, выигрыш можно получить только в задачах с большой вычислительной сложностью. Для получения более точной информации предполагается проведение аналогичных численных экспериментов для более сложных 2D и 3D задач.

БЛАГОДАРНОСТИ

Данная работа выполнена при поддержке Российского научного фонда (проект 18-11-00215).

СПИСОК ЛИТЕРАТУРЫ

1. *Reed D.A., Dongarra J.* Exascale Computing and Big Data, Communications of the ACM. 2015. V. 58. № 7. P. 56–68.
2. DVM-система, URL: http://dvm-system.org.
3. *Бахтин В.А., Клинов М.С., Крюков В.А., Поддерюгина Н.В., Притула М.Н., Сазанов Ю.Л.* Расширение DVM-модели параллельного программирования для кластеров с гетерогенными узлами // Супервычисления и математическое моделирование. Труды XIII Международного семинара / Под ред. Р.М. Шагалиева. – Саров: ФГУП “РФЯЦ-ВНИИЭФ”, 2012. С. 84–91.
4. *Бахтин В.А., Клинов М.С., Крюков В.А., Поддерюгина Н.В., Притула М.Н., Смирнов А.А.* Использование языка Fortran DVMH для решения задач гидродинамики на высокопроизводительных гибридных вычислительных системах // Вестник Южно-Уральского государственного университета, серия “Вычислительная математика и информатика”. 2013. Т. 2. № 3. С. 106–120.
5. *Бахтин В.А., Жукова О.Ф., Катаев Н.А., Колганов А.С., Крюков В.А., Поддерюгина Н.В., Притула М.Н., Савицкая О.А., Смирнов А.А.* Автоматизация распараллеливания программных комплексов // Научный сервис в сети Интернет: труды XVIII Всероссийской научной конференции (19–24 сентября 2016 г., г. Новороссийск). М.: ИПМ им. М.В. Келдыша, 2016. С. 76–85.
6. *Алексахин В.Ф., Бахтин В.А., Захаров Д.А., Колганов А.С., Королев А.В., Крюков В.А., Поддерюгина Н.В., Притула М.Н.* Опыт решения прикладных задач с использованием DVM-системы // Труды международной конференции Суперкомпьютерные дни в России (25–26 сентября 2017 г., г. Москва). М.: Изд-во МГУ, 2017. С. 650–661.
7. *Bakhtin V. A., Krukov V. A.* DVM-Approach to the Automation of the Development of Parallel Programs for Clusters Programming and Computer Software. 2019. V. 45. № 3. P. 121–132. https://doi.org/10.1134/S0361768819030034
8. *Bakhtin V.A., Zaharov D.A., Kolganov A.S., Krukov V.A., Podderyugina N.V., Pritula M.N.* Development of parallel applications using DVM-system // Vestnik Yuzhno-Ural’skogo Gosudarstvennogo Universiteta. Seriya “Vychislitelnaya Matematika i Informatika. 2019. V. 8. № 1. P. 89–106.
9. *Bondarev A.E.* On the Construction of the Generalized Numerical Experiment in Fluid Dynamics // Mathematica Montisnigri. 2018. V. XLII. P. 52–64.
10. *Bondarev A.E.* On visualization problems in a generalized computational experiment // Scientific Visualization. 2019. V. 11. № 2. P. 156–162.
11. *Bondarev A.E., Galaktionov V.A.* Generalized Computational Experiment and Visual Analysis of Multidimensional Data // Scientific Visualization. 2019. V. 11. № 4. P. 102–114.
12. *Bondarev A.E., Galaktionov V.A.* Parametric Optimizing Analysis of Unsteady Structures and Visualization of Multidimensional Data // International Journal of Modeling, Simulation and Scientific Computing. 2013. V. 4. P. 13.
13. *Bondarev A.E., Galaktionov V.A.* Analysis of Space-Time Structures Appearance for Non-Stationary CFD Problems // Procedia Computer Science. 2015. V. 51. P. 1801–1810.
14. *Bondarev A.E., Galaktionov V.A.* Multidimensional data analysis and visualization for time-dependent CFD problems // Programming and Computer Software. 2015. V. 41. № 5. P. 247–252. https://doi.org/10.1134/S0361768815050023
15. Разработка инструментального программного средства Burgers2 для оптимизации гибридных разностных схем / Бондарев А.Е. и др. Препринты ИПМ им. М.В. Келдыша. 2012. № 53. 12 с.

16. *Bondarev A.E.* On hybrid numerical method for 2d viscous flows, Mathematica Montisnigri. 2014. V. XXIX. P. 59–67.
17. *Sod G.A.* Survey of Several Finite Difference Methods for Systems of Nonlinear Hyperbolic Conservation Laws // Jurnal Comput. Phys. 1978. V. 27. P. 1–31.
18. *Toro E. F.* Riemann Solvers and Numerical Methods for Fluid Dynamics, Springer-Verlag, 1999. 624 p.
19. *Liu X.-D., Osher S., Chan T.* Weighted essentially non-oscillatory schemes // Journal of Computational Physics. 1994. V. 115. P. 200–212.
20. *Shu C.-W.* High order weighted essentially non-oscillatory schemes for convection dominated problems // SIAM Review. 2009. V. 51. P. 82–126.

ПРОГРАММИРОВАНИЕ, 2020, № 3, с. 24–34

КОМПЬЮТЕРНАЯ ГРАФИКА И ВИЗУАЛИЗАЦИЯ

УДК 004.92

ВОССТАНОВЛЕНИЕ ПАРАМЕТРОВ ОСВЕЩЕНИЯ В СИСТЕМАХ СМЕШАННОЙ РЕАЛЬНОСТИ С ПОМОЩЬЮ ТЕХНОЛОГИИ СВЕРТОЧНЫХ НЕЙРОННЫХ СЕТЕЙ ПО RGBD-ИЗОБРАЖЕНИЯМ

© 2020 г. М. И. Сорокин[a,*], Д. Д. Жданов[a,], А. Д. Жданов[a,***], И. С. Потемин[a,****], Н. Н. Богданов [a,*****]**

[a] *Санкт-Петербургский национальный исследовательский университет информационных технологий, механики и оптики*
197101 Санкт-Петербург, Кронверкский пр., 49, Россия

**E-mail: vergotten@gmail.com*

***E-mail: ddzhdanov@mail.ru*

****E-mail: adzhdanov@itmo.ru*

*****E-mail: ipotemin@yandex.ru*

******E-mail: bogdanov@k-remonta.com*

Поступила в редакцию 18.12.2019 г.
После доработки 10.01.2020 г.
Принята к публикации 21.01.2020 г.

Одной из основных проблем устройств смешанной реальности является отсутствие универсальных методов и алгоритмов, позволяющих естественным образом визуализировать объекты виртуального мира в реальном пространстве. Ключевым моментом естественного восприятия виртуальных объектов в реальном мире является создание естественных условий освещения объектов виртуального мира источниками света, находящимися в реальном мире, т.е. формирование естественных бликов на виртуальных объектах и теней от этих объектов в реальном мире. В работе предлагается метод адекватного определения положения основных источников реального мира в системах смешанной реальности. Современные технологии, объединяющие возможности формирования двухсполовиноймерных изображений, создаваемых камерами глубины, и их последующую компьютерную обработку с использованием нейронных сетей, позволяют выделить объекты реального мира, распознать их тени и корректно восстановить источники света, создающие эти тени. Приводятся результаты работы предложенного метода, оценивается точность восстановления положения источников света и демонстрируется визуальное отличие изображения сцены с исходными источниками света от этой же сцены с восстановленными параметрами источников света.

DOI: 10.31857/S0132347420030097

1. ВВЕДЕНИЕ

В настоящее время происходит быстрое развитие визуальных интерактивных технологий: VR (виртуальная реальность), AR (дополненная реальность), MR (смешанная реальность) [1–3]. На рынке представлено множество устройств VR, AR, MR [4–6], а над улучшением качества зрительного восприятия, включая решение проблемы восстановления параметров источников света реального мира, работают многие исследовательские группы [7–9]. Существует ряд реализованных пилотных проектов, использующих эти технологии не только в индустрии развлечений и игр. Технологии VR, AR, MR находят применение в медицине [10], в архитектуре [11], в военном деле [12] и в других областях человеческой деятельности.

Следует отметить, что технология смешанной реальности сложнее технологий виртуальной или дополненной реальностей. Если в системе виртуальной реальности человек полностью погружен в виртуальный мир и не видит окружающего пространства, а в системе дополненной реальности человек наблюдает вспомогательные элементы, такие как, например, данные навигации или информационные сообщения, то в системе смешанной реальности человек видит виртуальные объекты, внедренные в пространство реального мира. При этом человек должен воспринимать виртуальные объекты как объекты реального мира и у

него не должно возникать конфликта зрительного восприятия, вызванного, например, неестественным освещением виртуальных объектов. Для того, чтобы виртуальный объект выглядел достаточно реалистично, необходимо выполнить несколько условий. Во-первых, виртуальный объект должен иметь естественные оптические свойства поверхности (отражение, пропускание, преломление, включая соответствующие текстуры). В большинстве случаев свойства поверхностей виртуальных объектов (текстур и оптических свойств) могут быть назначены вычислительной системой (на основе данных из библиотек материалов, например) или пользователем (на основе личного опыта). Поэтому корректное назначение оптических свойств не является сложной проблемой

Вторым важным условием реалистичного восприятия виртуальных объектов является физическая корректность их освещения. То есть блики, яркие и затененные области на объектах должны соответствовать условиям реального освещения, а также виртуальные объекты должны отбрасывать тени в соответствии с этими условиями освещения. С точки зрения зрительного восприятия блики на виртуальных объектах и тени, которые они отбрасывают, должны коррелировать с бликами и тенями от реальных объектов и не вызывать чувства дискомфорта при наблюдении смешанного изображения.

В данной статье авторы предлагают эффективный метод восстановления положения источников света (ИС) реального мира по двухсполовиноймерным изображениям, получаемым с помощью камер глубины, которые могут быть интегрированы в системы смешанной реальности.

2. АНАЛИЗ СУЩЕСТВУЮЩИХ РАБОТ

Ранее авторы рассматривали вопрос восстановления распределения яркости методом трех сфер, который заключался в определении координат положения источника света в пространстве сцены [13]. В статье было показано, что разработанный метод определения координат положения источника света на основе анализа HDRI может обеспечить достаточно точные результаты. Тем не менее, метод имеет ряд недостатков, таких как:

1. Метод работоспособен только при условии, что, во-первых, поверхности имеют свойства отражения, близкие к закону Ламберта, то есть идеально “матовые” или диффузно отражающие поверхности, и во-вторых, диаграмма излучения источника света близка к закону Ламберта.

2. Для локальной области изображения метод позволяет найти только один источник света.

Предлагаемый метод лишен этих недостатков. Он может работать с произвольными диффузными поверхностями (заданными ДФР), а также с несколькими источниками света.

Вопросами восстановления условий освещения реальных сцен занимаются различные группы исследователей. В работе [14] представлен метод, основанный на высококачественной оценке освещенности методами сверточной нейронной сети (СНС, англ. – CNN). Авторы обучают СНС, используя синтезированные изображения, и впоследствии применяют ее для анализа реальных изображений. Чтобы поддерживать точность и эффективность метода, результаты оценки освещенности объединяются от нескольких экземпляров СНС. Экспериментальные результаты показывают, что предлагаемый метод дает достаточно точные оценки при анализе изображений реального мира.

В работе [15] представлен метод визуализации теней с помощью освещения, полученного в ходе анализа изображений в приложениях дополненной реальности. Для аппроксимации результатов освещения и затенения окружающей среды система использует купол с разноцветными источниками света. Цвет каждой тени определяется областью окружающей среды за источником света. В результате становится возможным устанавливать непосредственное влияние изменений условий освещения на отбрасывание теней виртуальными объектами.

В статье [16] предложена концепция анализа теней в режиме реального времени для приложений дополненной реальности, использующих теневые объемы. Концепция была реализована в прототипе “shadowAReality” с положительными результатами. Тени значительно улучшают реальную сцену и предлагают пользователю более интуитивный и реалистичный мир. Как представлено в статье [17], алгоритм анализа теневого объема может быть улучшен за счет использования порталов, окклюзии и методов отбора усеченного вида, позволяющих избежать рендеринга ненужных теневых объемов. Кроме того, возможно улучшение алгоритма теневого объема с использованием языка затенения nVIDIA Cg [18, 19].

Авторы работы [20] представляют метод восстановления освещения и свойств поверхности по случайно отсканированной геометрии. Это означает быструю и потенциально “шумную” процедуру сканирования немодифицированных и неструктурированных сцен с помощью стандартного датчика RGB-D. В отличие от процедур восстановления светотехнических характеристик объектов, требующих тщательной подготовки в лабораторных условиях, этот метод работает с

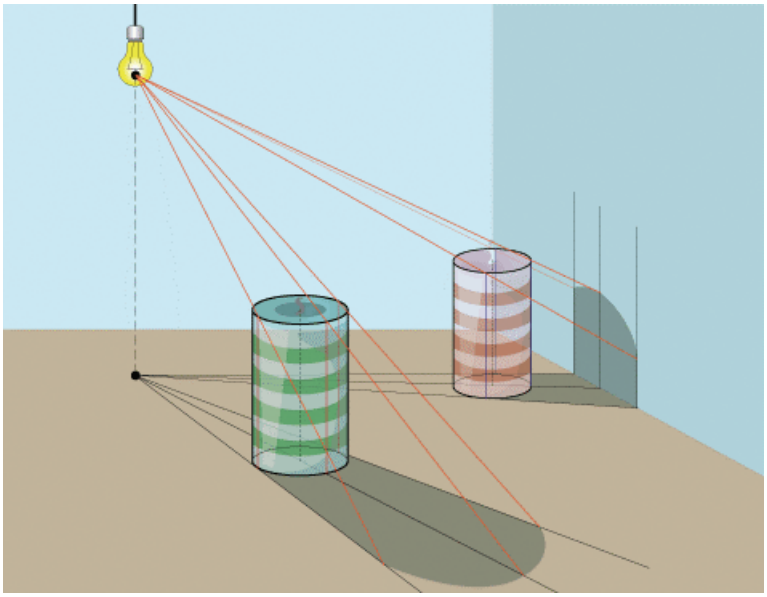


Рис. 1. Метод восстановления координат источников света по теням от объектов.

данными, которые могут быть получены пользователями в полевых условиях. Чтобы обеспечить надежную процедуру восстановления, авторы сегментировали полученную геометрию на поверхности с однородными свойствами материала и рассчитали перенос излучения на этих сегментах. С такими входными данными авторы решили обратную задачу рендеринга – факторизацию освещения и свойств материала, используя итеративную оптимизацию в форме сферических гармоник. Это позволяет учитывать самозатенение и восстанавливать зеркальные свойства объектов. Полученные данные можно использовать для генерации широкого спектра приложений смешанной реальности, включая рендеринг синтетических объектов с соответствующим освещением в заданной сцене, а также синтез изображения сцены (или ее части) с новым освещением. Была продемонстрирована надежность данного подхода на реальных и синтезированных примерах в различных условиях освещения и проведено сравнение с исходными данными.

Для восстановления параметров естественного и искусственного освещения по HDRI изображениям существует ряд работ [21–23], которые позволяют найти и выделить яркие источники света, создающие блики и тени. Однако для гарнитур смешанной реальности при нахождении параметров естественного освещения (положение солнца) более естественный подход должен основываться на анализе параметров датчиков, определяющих ориентацию гарнитуры в пространстве, дате и времени, привязывая эти параметры к параметрам модели неба и, соответственно, положению солнца на небе.

В работе [24] предлагается метод восстановления контуров теней с помощью ROI (Region of interest) и анализа прилегающих пикселей объекта на изменение контрастности. Метод достаточно простой и надежный, но для простых объектов и теней. В отличие от данного метода сверточные сети позволяют определять сложные тени по всему изображению, а не только в интересующей области.

Рассмотренные работы предлагают эффективные подходы для восстановления распределения освещения в системах смешанной реальности, что в ряде случаев позволяет получить удовлетворительные результаты. С другой стороны, рассмотренные подходы хорошо работают, когда в сцене присутствует только один источник света (солнце или искусственный источник света) или источники света находятся на значительном расстоянии (практической бесконечности). В реальном мире присутствует множество источников света. Поэтому авторами предложен новый подход, который может восстановить параметры освещения в сложных сценах с несколькими источниками света.

3. МЕТОД

В настоящей работе предложен метод, который заключается в определении источников освещения сцены по RGBD-изображению сцены. Данное RGBD-изображение позволяет получить координаты точек теней и объектов сцены,

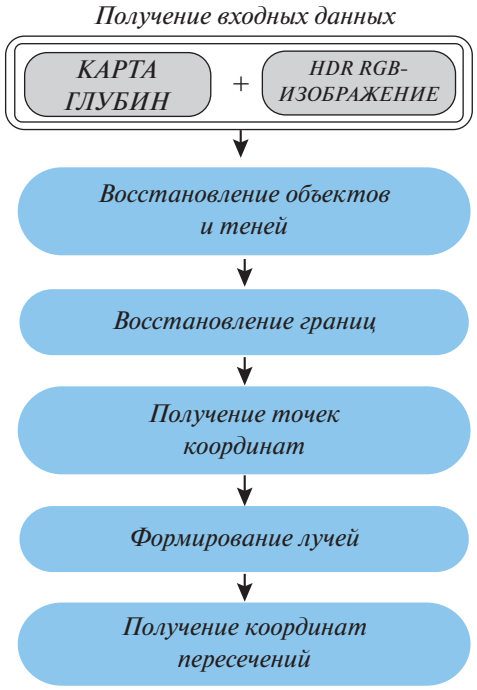


Рис. 2. Алгоритм метода восстановления координат источников света.

создающих эти тени. Метод основывается на формировании и анализе пучков лучей, соединяющих координаты объекта и тени. Основное положение, на котором базируется данный метод, заключается в том, что для каждой точки на границе тени существует точка на объекте, которая соединит его с источником света. Поэтому, в области максимальной концентрации "пересечений" лучей, идущих от точек, находящихся на границах теней, на точки, находящиеся на границах объектов, будет находиться источник света. Кроме того, по размеру области концентрации лучей можно оценить размер источника света. Данный метод позволяет работать как со сценами, в которых источник света может отсутствовать в поле зрения, так и со сценами, содержащими большое количество источников света, включая неточечные источники света. Естественно, предложенный метод может обнаружить только те источники света, которые отбрасывают различимые тени.

Следует отметить, что предложенный метод работает не с 3D моделями сцены, а с изображениями и картами глубин, которые могут быть получены с использованием специальных устройств, например, 3D сканеров или лидаров, которые могут определить расстояние до любой точки изображения и формируют эту информацию значительно быстрее, чем создание полной 3D модели.

На рисунке 1 наглядно представлено изображение метода восстановления координат источников света по координатам объектов и их теней.

Алгоритм предлагаемого метода восстановления координат источников света представлен на рисунке 2 и состоит из следующих этапов:

1. Получение входных данных от устройства MR: карта глубины и соответствующее изображение видимой области сцены.

2. С помощью сверточной нейронной сети определение всех теневых участков изображения и нахождение их границ.

3. Используя пространственный фильтр и алгоритм фильтрации Canny в ROI, определение объектов, отбрасывающих тени, и выделение их разными цветами. Нахождение границ объектов.

4. Сохранение координат точек границ объектов и теней.

5. Формирование пучков лучей. Пучки лучей испускаются через координаты точек границы тени на точки на границе объектов. Точки на границе тени выбираются с шагом 5 пикселей и из каждой точки на границе тени с шагом 5 пикселей обстреливается граница объекта.

6. Поиск области "пересечения" лучей (каустики лучей) от различных точек на границе тени. Считается, что источник может находиться там, где есть "пересечение" как минимум трех лучей.

Реализация данного метода графически проиллюстрирована на рисунке 3, где одним цветом выделены лучи, проходящие через одну точку на границе объекта от различных точек на границе тени, а точкой выделены координаты с наибольшей плотностью каустики, то есть наиболее вероятным положением источника света.

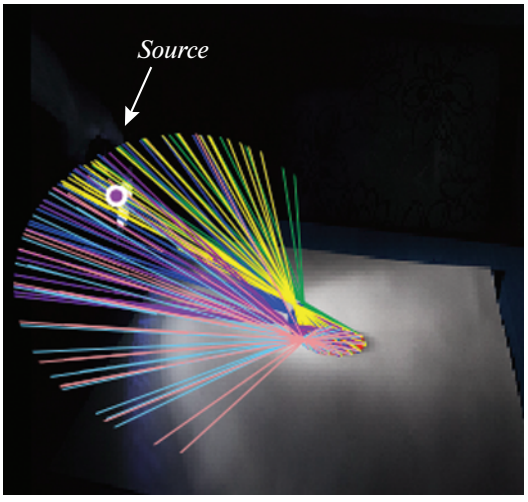


Рис. 3. Иллюстрация метода восстановления координат источников света.

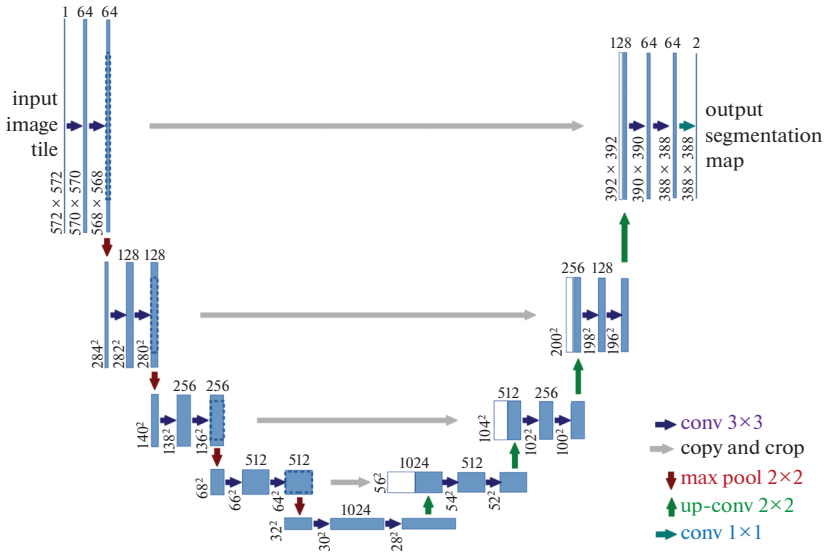


Рис. 4. U-Net архитектура нейронной сети.

Для восстановления координат точек объекта используется подход ROI (Region of Interest) относительно изображения тени. Если предположить, что объект соприкасается с тенью (это характерно для большинства объектов и их теней), то алгоритм обрабатывает эти участки (ROI) с помощью фильтра “Canny” и вычитает уже известные координаты теней, тем самым оставляя только координаты объекта. В фильтр “Canny” так же входит сглаживание по Гауссу для поиска градиентов и устранения шума.

Использование данного подхода позволяет отделить области тени от самого объекта, а использование функций библиотеки “NumPy” дает возможность получить координаты всех ненулевых пикселей отдельно для тени и для объекта.

4. РЕАЛИЗАЦИЯ

В основе предлагаемого метода лежит полносверточная нейронная сеть, позволяющая на исходном изображении сцены выделить тени и объекты, их формирующие, а также алгоритмы восстановления источников света по двухсполовиноймерным изображениям объектов и их теней. В качестве тренировочного набора данных был использован набор данных “SBU_Shadow” [25], который состоит из оригинальных изображений и их масок, где белым цветом выделена тень, а черным – незатененные участки.

В качестве архитектуры полносверточной сети было решено взять архитектуру U-Net, поскольку она наиболее хорошо подходит для работы с бинарной классификацией.

Архитектура состоит из 4 блоков “downsample” слоев для обучения классификации, 4 блоков “upsample” слоев, для получения на выходе массива той же размерности, что и на входе, а также “bottleneck” блоков со значением карты признаков 256 для более глубокого обучения сети.

Архитектура U-NET представлена на рисунке 4.

Пример пары изображений данного набора представлен на рисунке 5.

Тренировочный набор данных состоит из 4085 пар изображений, а тестовый из 638. Для схождения нейронной сети потребовалось всего 6 эпох,

Рис. 5. Пример пары изображений данных, используемых для обучения нейронной сети.

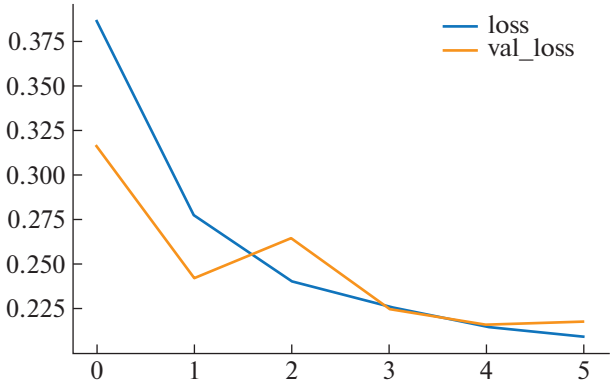


Рис. 6. История обучения нейронной сети.

что заняло на GeForce 1080Ti порядка 3 минуты. Скорость работы уже обученной нейронной сети составляет 28 мс.

В качестве функций активации слоев использовалась функция sigmoid, поскольку по своей природе она нелинейна, а комбинация таких функций производит также нелинейную функцию. Еще одно достоинство функции sigmoid это то, что она гладкая, и в отличие от ступенчатой функции она позволяет сделать активацию аналоговой. Для сигмоиды также характерен гладкий градиент. Сигмоида выглядит подходящей функцией для задач классификации. Она стремится привести значения к одной из сторон кривой (например, к верхнему при $x = 1$ или нижнему при $x = -1$ пределу). Такое поведение позволяет находить четкие границы при предсказании. Из недостатков же стоит отметить то, что при приближении к концам сигмоиды значения Y имеют тенденцию слабо реагировать на изменения в X. Это означает, что градиент в таких областях принимает маленькие значения. А это, в свою очередь, приводит к проблемам с градиентом исчезновения.

На рисунке 6 изображена история обучения нейронной сети в виде графика, где по горизонтали отображены эпохи, а по вертикали – уменьшение потерь. Данный график построен для обучающихся данных (loss) и для проверочных (val_loss). Задача нейронной сети максимально снизить ошибку целевой функции, чего она и достигает на 5 эпохе со значениями loss: 0.2093 и val_loss: 0.2179. В качестве функции ошибки в данной работе использовалась 'binary_crossentropy', а в качестве оптимизатора (функция обновления весов) – RMSprop со значениями lr=1e-4, decay=1e-6.

Точность классификации на проверочных данных составила 92 процента, а на тестовых – 94 процента, используя IoU (intersection over union) метрику. Как можно увидеть на рисунке 7, результаты работы нейронной сети очень близки к эталонным. Левые изображения на рисунке 7 – оригинальные изображения, которые подаются на вход нейронной сети, правые – эталонные или “ground truth”, а посередине – результат предсказания нейронной сети.

После того, как определили участки с тенью на изображении, берется ROI этой области и применяется оператор Canny. Зная координаты области с тенью, убираются все внутренние и наружные пиксели и остаются только области с контурами самого объекта.

Результат работы оператора “Canny” представлен на рисунке 8.

Зная координаты объекта и тени, испускаются пучки лучей, соединяющие точки на контуре тени с точками на контуре объекта. Данный алгоритм детально описан в предыдущей главе. Функция, хранящая координаты всех испущенных лучей, реализована в виде numpy-массива. Библиотека numpy выбрана из-за удобства и эффективности работы с данными. Затем для групп лучей, выпущенных с разных точек на границе тени, находятся области каустик, соответствующие максимальной близости лучей. Эти области являются источниками света, освещающего сцену.

Данный метод позволяет определить источники освещения в заданной системе координат. В простейшем случае система координат формируется относительно текущего изображения, т.е. одна из осей ориентирована по направлению на центр изображения, а две другие – по осям экрана. Использование данного метода с 3D сканером или гарнитурой смешанной реальности, позиционирующей свое положение в пространстве, позволит определить координаты источников света в пространстве сцены.

На рисунке 9 показан пример работы данного алгоритма на примере пяти реальных изображений. Слева направо приведены исходные изображения и промежуточные результаты работы алгоритмов по определению координат объектов и их теней на изображении. На рисунке представлены: оригинальные изображения, изображения, полученные с помощью фильтра Canny, изображения с тенями и изображения с объектами. Имея данные изображения, можно определить координаты на объектах и тенях изображений, затем построить лучи с границ теней на границы объектов и определить источники света. Сцены пронумерованы по порядку, от первой сверху до пятой внизу рисунка.

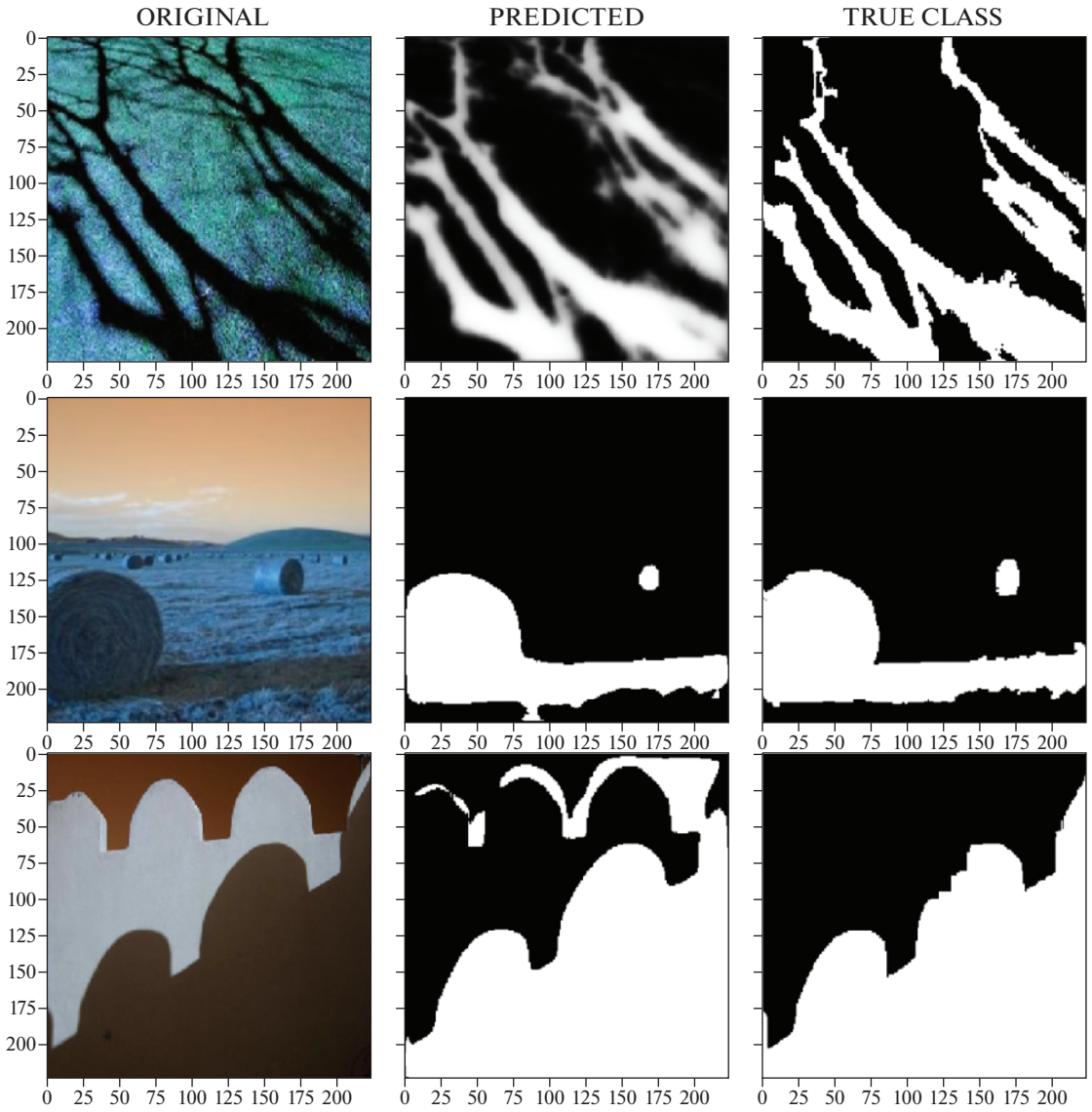


Рис. 7. Результаты работы нейронной сети.

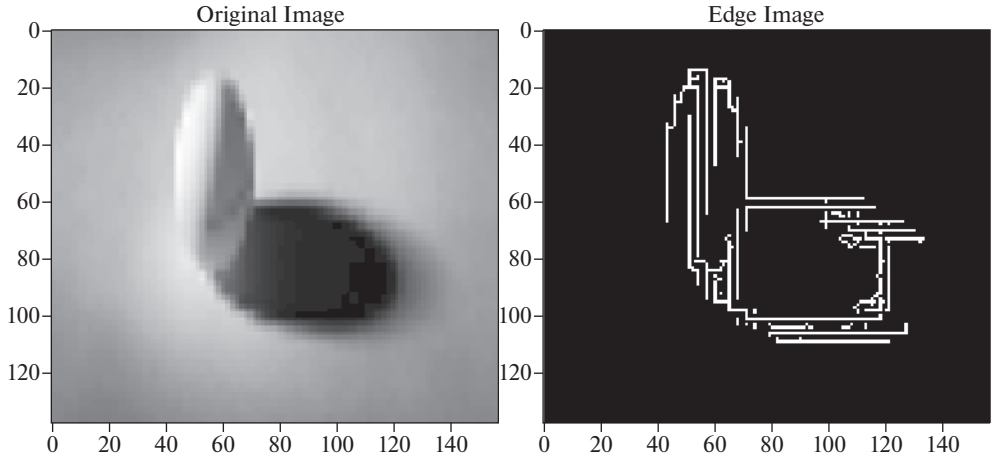


Рис. 8. Результат работы оператора Canny для выделения контуров.

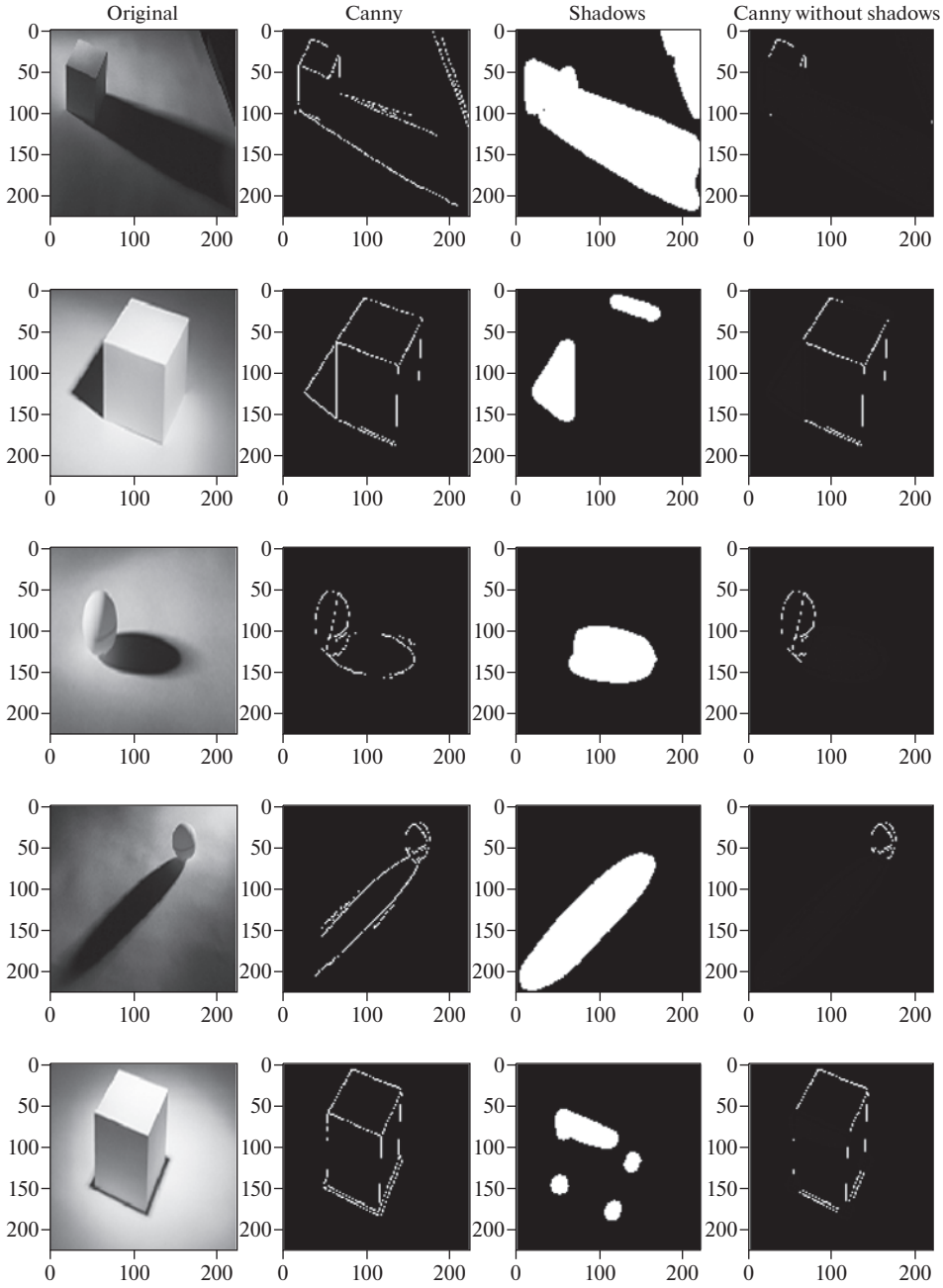


Рис. 9. Пример работы алгоритмов по определению координат границ объектов и теней.

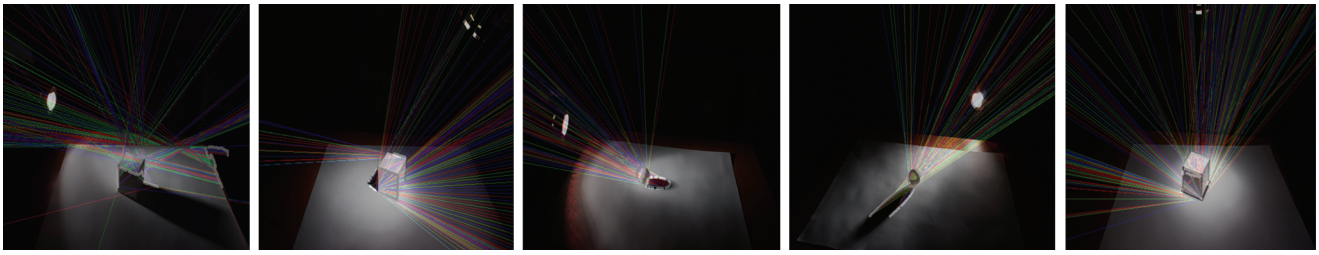


Рис. 10. Результат работы алгоритма по определению источников освещения (1–5 сцена слева направо).

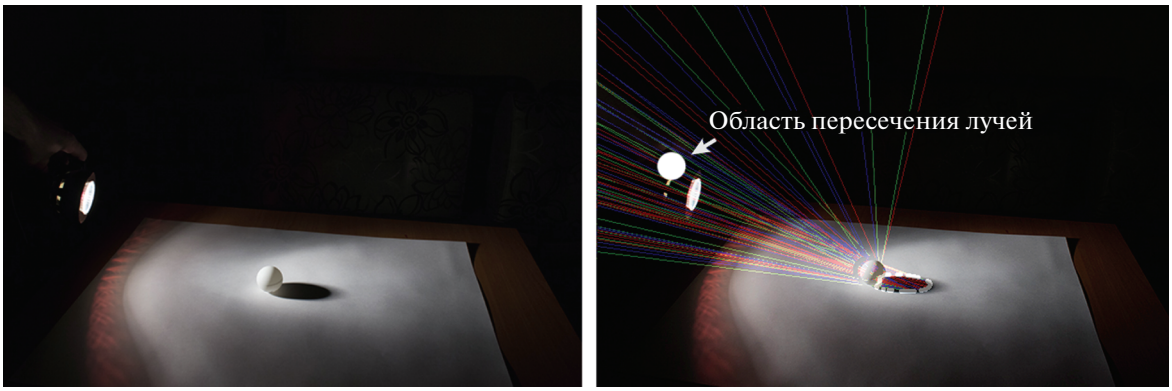


Рис. 11. Область пересечения лучей сцены относительно исходного источника света.

На рисунке 10 визуализирован результат работы алгоритма по определению координат источников света с помощью анализа трасс лучей.

Более подробно результат работы алгоритма для третьей сцены представлен на рисунке 11. Слева приведено оригинальное изображение, а справа результат работы с выделенной областью пересечения лучей. Абсолютная погрешность относительно источника освещения составляет 0.109 метра (или 37 пикселей экрана), относительная погрешность – 13.9%, угловая погрешность ориентации источника света относительно центра освещаемого объекта сцены – 0.0606 радиана.

В таблице 1 представлены результаты восстановления координат источников света для пяти сцен, приведенных выше. Погрешности восстановления координат источника света приводятся в абсолютном виде как отклонение центра исходного источника от центра восстановленного источника света (в пространстве реальной сцены – в метрах и в пространстве экрана – в пикселях) и как отклонение между направлениями от центра объекта на исходный источник света и от центра объекта на восстановленный источник света – в радианах. Кроме того, приводится относительная ошибка восстановления источника света как разность между положениями центров исходного и восстановленного источников света, приведенная к расстоянию от центра объекта до исходного источника света – в процентах.

Размеры оригинальных изображений сцен составляют 622 × 415 пикселей. Размеры ROI участков изображения – 224 × 224 пикселя, что дает возможность запустить их на вход нейронной сети и классифицировать теневые области.

Все операции были выполнены на компьютере с процессором Ryzen7-1700 и видеокартой GTX 1080Ti. Время распознавания тени на изображении составило 32 ms, распознавание объекта – 25 ms и поиск координат положения центра источника света методом поиска области “пересечения” лучей – 875 ms.

В данной работе использовался язык программирования Python и библиотеки OpenCV, Keras, Numpy и Scikit-learn.

Таблица 1. Точность определения источников света

Сцена, n	Абсолютная погрешность (в пикселях)	Абсолютная погрешность (в метрах)	Относительная погрешность	Угловая погрешность (в радианах)
1	12	0.036	2.58%	0.0004
2	26	0.077	8.1%	0.0084
3	37	0.109	13.9%	0.0606
4	42	0.124	13.6%	0.0412
5	19	0.056	6.7%	0.0334

5. ЗАКЛЮЧЕНИЕ

В данной работе показано, что разработанный метод подходит для работы с системами дополненной реальности и позволяет решить задачу восстановления координат центров источников света в системе координат сцены. В работе использовалась сверточная нейронная сеть с архитектурой U-Net, после обучения которой точность классификации составила 94 процента. Архитектура данной сети подходит для бинарной классификации данных и может распознавать сложные тени на изображениях. Скорость работы нейронной сети позволяет использовать данное решение в системах реального времени, а скорость восстановления параметров источника света лучевым методом, после его реализации на GPU, также позволит использовать предложенный подход в системах реального времени. Точность восстановления координат центров источников света в большинстве случаев была достаточной для формирования естественного визуального восприятия освещения виртуальных объектов источниками света реального мира. В дальнейшем планируется повысить точность восстановления координат источников света, особенно в случаях сложного освещения, и реализовать возможность оценки размера и формы протяженного источника света.

БЛАГОДАРНОСТИ

Работа выполнена при финансовой поддержке Российского научного фонда (проект № 18-79-10190).

СПИСОК ЛИТЕРАТУРЫ

1. *Parsons S., Cobb, S.* State-of-the-art of virtual reality technologies for children on the autism spectrum // European Journal of Special Needs Education. 2011. V. 26. № 3. P. 355–366.
2. *Palmarini R.* A systematic review of augmented reality applications in maintenance // Robotics and ComputerIntegrated Manufacturing. 2018. V. 49. P. 215–228.
3. *Izadi S.* The Reality of Mixed Reality // Proceedings of the 2016 Symposium on Spatial User Interaction, 2016, ACM. P. 1–2.
4. Oculus Rift, https://www.oculus.com (10 April 2019).
5. Epson Moverio, https://moverio.epson.com (10 April 2019).
6. Microsoft Hololens, https://www.microsoft.com/en-us/hololens (10 April 2019).
7. *William R., Craig A.* Understanding virtual reality: Interface, application, and design, Morgan Kaufmann, 2018.
8. *Iis T. P., Jung T. H., Claudia M. D.* Embodiment of wearable augmented reality technology in tourism experiences // Journal of Travel research. 2018. V. 57. № 5. P. 597–611.
9. *Trout T.* Collaborative mixed reality (MxR) and networked decision making // Next-Generation Analyst VI. V. 10653. International Society for Optics and Photonics, 2018.
10. *Sheena B., Anandasabapathy S., Shukl, R.* Use of augmented reality and virtual reality technologies in endoscopic training // Clinical Gastroenterology and Hepatology. 2018. V. 16. № 11. P. 1688–1691.
11. *Kiljae A., Ko D., Gim S.* A Study on the Architecture of Mixed Reality Application for Architectural Design Collaboration // International Conference on Applied Computing and Information Technology. Springer, Cham. 2018. P. 48–61.
12. *Livingston M., Zhuming A., Decker J. W.* Human Factors for Military Applications of Head-Worn Augmented Reality Displays // International Conference on Applied Human Factors and Ergonomics. Springer, Cham. 2018. P. 56–65.
13. *Wang X., Zhdanov D. D., Potemin I. S., Wang Y., Cheng, H.* The efficient model to define a single light source position by use of high dynamic range image of 3D scene // The international society for optical engineering, Optoelectronic Imaging and Multimedia Technology IV, 100200I (31 October 2016).
14. *Mandl D., Yi K. M., Mohr P., Roth P. M., Fua P., Lepetit V., Schmalstieg D., Kalkofen D.* Learning lightprobes for mixed reality illumination // IEEE International Symposium on Mixed and Augmented Reality (ISMAR). IEEE. 2017. P. 82–89.
15. *Supan P., Stuppacher I., Haller M.* Image Based Shadowing in Real-Time Augmented Reality // International Journal of Virtual Reality. 2006. V. 5. № 3. P. 1–7.
16. *Haller M., Drab S., Hartmann W.* A real-time shadow approach for an augmented reality application using shadow volumes // Proceedings of the ACM symposium on Virtual reality software and technology. ACM. 2003. P. 56–65.
17. *Everitt C., Kilgard M. J.* Practical and Robust Stenciled Shadow Volumes Hardware-Accelerated Rendering, arXiv preprint cs/0301002, 2003.
18. *Randima F., Kilgard M. J.* The Cg Tutorial: The definitive guide to programmable real-time graphics, Addison-Wesley Longman Publishing Co., Inc., 2003.
19. *Kirk D. CG* Toolkit, User's Manual, Nvidia Corporation, Santa Clara, CA, 2002.
20. *Richter-Trummer T.* Instant mixed reality lighting from casual scanning // IEEE International Symposium on Mixed and Augmented Reality (ISMAR). IEEE, 2016. P. 27–36.
21. *Волобой А.Г., Галактионов В.А., Копылов Э.А., Шапиро Л.З.* Расчет солнечного освещения, заданного изображением с большим динамическим диапазоном // Труды16-ой международной конференции по компьютерной графике и ее приложениям – ГрафиКон'2006, Россия, Новосибирск, июль 1–5, 2006. С. 467–472.
22. *Волобой А.Г., Галактионов В.А., Копылов Э.А., Шапиро Л.З.* Моделирование естественного дневного

освещения, задаваемого изображением с большим динамическим диапазоном // “Программирование”, № 5, 2006. С. 62–80.

23. *Валиев И.В., Волобой А.Г., Галактионов В.А.* Физически корректная модель солнечного освещения, задаваемая изображением с большим динамическим диапазоном // “Вестник компьютерных и информационных технологий”, № 9, 2009. С. 10–17.
24. *Jiddi S., Robert P., Marchand E.* Illumination Estimation using Cast Shadows for Re- alistic Augmented Reality Applications // IEEE Int. Symposium on Mixed and Augmented Reality (ISMAR-Adjunct), Oct 2017, Nantes, France. 2017.
25. *Vicente T. F. Y., Hou L., Yu C.-P., Hoai M., Samaras D.* Large-scale training of shadow detectors with noisily-annotated shadow examples // Proceedings of the European Conference on Computer Vision, 2016.

ПРОГРАММИРОВАНИЕ, 2020, № 3, с. 35–41

КОМПЬЮТЕРНАЯ ГРАФИКА И ВИЗУАЛИЗАЦИЯ

УДК 004.932.4

ПРОЕКЦИОННЫЙ МЕТОД АНАЛИЗА ПЕРФУЗИОННЫХ ИЗОБРАЖЕНИЙ МОЗГА

© 2020 г. Д. А. Люков[a,b,*], А. С. Крылов[a,b,], В. А. Лукшин[c,***], Д. Ю. Усачев[c]**

[a] *Московский государственный университет имени М.В. Ломоносова*
Факультет вычислительной математики и кибернетики
Лаборатория математических методов обработки изображений
119991 Москва, ГСП-1, Ленинские горы, д. 1, Россия

[b] *Московский Центр фундаментальной и прикладной математики*
119991 Москва, ГСП-1, Ленинские горы, д. 1, Россия

[c] *Национальный медицинский исследовательский центр нейрохирургии имени академика Н.Н. Бурденко*
125047 Москва, 4-я Тверская-Ямская, д. 16, Россия

**E-mail: d@lyukov.com*

***E-mail: kryl@cs.msu.ru*

****E-mail: wlukshin@nsi.ru*

Поступила в редакцию 25.12.2019 г.
После доработки 09.01.2020 г.
Принята к публикации 13.01.2020 г.

Количественное исследование движения крови в тканях головного мозга – одна из актуальных задач в нейрохирургии. В частности, она возникает при диагностике острого ишемического инсульта. Эта задача решается, в том числе, с помощью перфузионной компьютерной томографии. Существуют различные методы извлечения количественных характеристик мозгового кровотока из данных перфузионной компьютерной томографии, отличающиеся разной степенью устойчивости к шуму. Более устойчивые к шуму методы позволяют использовать меньшие дозы радиации при проведении исследования пациента. Поэтому создание устойчивых методов является важной задачей. В данной работе представлен алгоритм рассчета количественных характеристик мозгового кровотока, основанный на регуляризации с помощью проекции на множество монотонных функций в комбинации с минимизацией функционала обобщенной полной вариации. Проведено тестирование предложенного подхода на синтетических и реальных данных. Предложенный метод показал результаты лучшие, чем метод сингулярного разложения с регуляризацией Тихонова, метод минимизации полной вариации и метод минимизации обобщенной полной вариации.

DOI: 10.31857/S013234742003005X

1. ВВЕДЕНИЕ

Ишемический инсульт – одна из главных причин смертности во всем мире. Одна из задач, возникающих в нейрохирургии – это локализация участка мозга, пораженного инсультом. Количественно оценить движение крови в каждом элементе объема головного мозга позволяет перфузионная компьютерная томография. Данный метод заключается в введении в кровь пациента контрастного вещества и последующем измерении его концентрации в течение некоторого промежутка времени с помощью компьютерной томографии (КТ) [1].

Необходимо получить пространственные карты характеристик мозгового кровотока: среднюю скорость мозгового кровотока (cerebral blood flow, CBF), средний объем мозгового кровотока (cerebral blood volume, CBV), среднее время доставки крови (mean transit time, MTT) [2]. Затем эти пространственные карты используются для диагностики. При этом может также использоваться проекция данных пациента на атлас [3].

Схема задачи представлена на рис. 1.

Радиационное облучение от компьютерной томографии может приводить к некоторому вреду для пациента, как, например, повреждению кожи или волос, а также к увеличению риска онкологических заболеваний. Поэтому одной из целей является снижение дозы радиации во время исследования. Однако использование КТ с низкой радиационной дозой, в свою очередь, приводит к большой зашумленности КТ-изображений. Необходимо разрабатывать методы, которые способны показывать устойчивый результат даже при сильно зашумленных входных данных. Такой

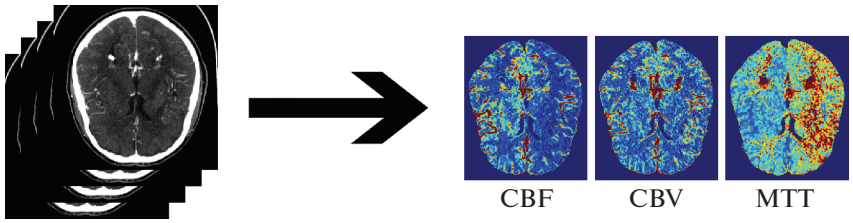


Рис. 1. Схема задачи. Входные данные — набор изображений КТ, показывающих изменение во времени. Необходимо оценить характеристики кровотока: CBF, CBV, MTT.

подход может помочь уменьшить дозу радиации в КТ-исследованиях.

Теоретической основой большинства методов является сверточная модель, описанная в разделе 2. Основная задача – это задача обращения свертки, то есть обратная задача для этой модели. Есть также методы, не использующие обращение свертки [4], но они показывают гораздо меньшую устойчивость к шуму.

Существует множество методов для решения этой задачи. К прямым методам относятся метод сингулярного разложения и различные способы его регуляризации, такие как метод сингулярного разложения с пороговой фильтрацией и регуляризация Тихонова [5, 6]. При использовании метода сингулярного разложения задача рассматривается в каждой точке независимо. Другой класс методов включает в себя итерационные методы. Один из них это метод минимизации функционала полной вариации [7, 8]. Этот метод более устойчив к шуму, чем метод сингулярного разложения с регуляризацией Тихонова, поскольку он использует также пространственные корреляции в данных. Однако подход, основанный на минимизации полной вариации, приводит к кусочно-постоянным артефактам. Эта проблема может быть решена с помощью минимизации обобщенной полной вариации. Также предложен метод [9], где минимизация полной вариации объединена с использованием низко-ранговой регуляризации.

Другой подход к повышению устойчивости методов заключается в подавлении шума во входных либо итоговых изображениях [10]. Однако появление шума в итоговых изображениях вызвано по большей части некорректностью задачи обращения свертки, поэтому разумным шагом является использование мощного регуляризатора при решении основной задачи, а не только для пред- или постобработки.

В работе [11] проведено статистическое исследование того, как каждый компонент системы перфузионной компьютерной томографии количественно влияет на точность параметрических карт перфузии, получаемых сверточными моделями.

Использование сверточных нейронных сетей для данной задачи на данном этапе относится к нейросетевой интерполяции серии КТ-изображений по ограниченному поднабору [12] и прямой оценке объема инфаркта по набору КТ-изображений, не осуществляя расчета пространственной карты характеристик мозгового кровотока [13].

В данной работе рассматривается улучшение метода минимизации обобщенной полной вариации благодаря использованию предположения о монотонности искомых функций по времени, ранее не использовавшегося при регуляризации решений для сверточных моделей КТ-перфузии.

2. ТЕОРЕТИЧЕСКАЯ МОДЕЛЬ

Исходными данными в задаче являются данные компьютерной томографии в различные моменты времени. Таким образом, изначально имеется четырехмерное изображение. Однако на практике одно из пространственных измерений очень разреженное по сравнению с остальными двумя. Поэтому мы рассматриваем только три размерности: две пространственные и время.

Интенсивность изображения рассматривается как концентрация контрастного вещества (КВ) $c(t; x, y)$. Одна из точек изображения выбирается как артерия. Концентрация КВ в этой точке в зависимости от времени будем называть артериальной функцией (artery input function, AIF):

$$AIF(t) = c(c; a, b), \tag{2.1}$$

где (a, b) – координаты артерии на изображении.

Концентрация КВ в точках ткани связана с артериальной функцией с помощью уравнения свертки [6]:

$$c(t; x, y) = (AIF * k)(t) = \int_{-\infty}^{+\infty} AIF(\xi)k(t-\xi; x, y)d\xi, \tag{2.2}$$

где $k(t; x, y)$ – остаточная функция в точке (x, y). В уравнении (2.2) мы предполагаем, что все функции обращаются в ноль при $t < 0$.

Все описанные в разделе 1 характеристики кровотока могут быть вычислены в точке (x, y) из значений остаточной функции $k(t; x, y)$ [6]:

$$CBF = \frac{1}{\rho_{tissue}} \max k(t;x,y),$$
$$CBV = \frac{1}{\rho_{tissue}} \int_0^\infty k(\tau;x,y)d\tau, \quad (2.3)$$
$$MTT = \frac{1}{\max k(t)} \int_0^\infty k(\tau;x,y)d\tau,$$

где ρ_{tissue} – плотность ткани, которую можно считать константой.

3. ОБРАЩЕНИЕ СВЕРТКИ

На практике функции $k(t)$ и $c(t)$ рассматриваются на конечной равномерной сетке. Таким образом, заменяя интегралы на конечные суммы, можно переписать уравнение (2.2) в виде системы линейных алгебраических уравнений:

$$\mathbf{Ak} = \mathbf{c}, \quad (3.1)$$

где

$$\mathbf{A} = \Delta t \begin{pmatrix} AIF(t_1) & 0 & \dots & 0 \\ AIF(t_2) & AIF(t_1) & \dots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ AIF(t_T) & AIF(t_{T-1}) & \dots & AIF(t_1) \end{pmatrix},$$

а $t_1, t_2, \dots, t_T$ – узлы равномерной сетки с шагом Δt.

Поиск $k(t)$ в уравнении (2.2) является обратной некорректной задачей, поэтому конечная аппроксимация (3.1) этой задачи имеет плохо обусловленную матрицу $\mathbf{A}$.

3.1. Сингулярное разложение матрицы (SVD)

Матрицу $\mathbf{A}$ можно представить в виде произведения трех матриц [14]:

$$\mathbf{A} = \mathbf{U}\Sigma\mathbf{V}^{\mathbf{T}}, \quad (3.2)$$

где $\mathbf{U}$ и $\mathbf{V}$ – ортогональные матрицы, а

$$\Sigma = diag[\sigma_1, \sigma_2, \dots, \sigma_r],$$

– диагональная матрица, составленная из сингулярных значений матрицы $\mathbf{A}$, $r = rang\mathbf{A}$. Это представление называется сингулярным разложением матрицы $\mathbf{A}$.

Используя сингулярное разложение, мы можем переписать решение системы (3.1) в виде:

$$\mathbf{k_{ls}} = \mathbf{V}\Sigma^{-1}\mathbf{U}^{\mathbf{T}}\mathbf{c}, \quad (3.3)$$

где $\Sigma^{-1} = diag[\sigma_1^{-1}, \sigma_2^{-1}, \dots, \sigma_r^{-1}]$.

Решение этой задачи является также решением следующей задачи минимизации:

$$\mathbf{k_{ls}} = \arg\min_{\mathbf{k}\in\mathbb{R}^T} (\|\mathbf{Ak} - \mathbf{c}\|_2^2). \quad (3.4)$$

Маленькие сингулярные значения вносят большой вклад в значения $\mathbf{k_{ls}}$. Это является причиной плохой обусловленности матрицы $\mathbf{A}$. Однако эти значения можно подавить с помощью сглаживающего множителя:

$$\sigma_{i,\lambda}^{(tikh)} = \frac{\sigma_i}{\sigma_i^2 + \lambda^2}. \quad (3.5)$$

Используя матрицу

$$\Sigma_\lambda^{-1} = diag[\sigma_{1,\lambda}^{(tikh)}, \sigma_{2,\lambda}^{(tikh)}, \dots, \sigma_{r,\lambda}^{(tikh)}],$$

вместо Σ^{-1} в (3.4), мы получим метод, называемый методом сингулярного разложения с регуляризацией Тихонова. Здесь λ – параметр регуляризации. Вектор $\mathbf{k}_\lambda^{(tikh)}$, полученный этим методом, является решением следующей задачи минимизации:

$$\mathbf{k}_\lambda^{(tikh)} = \arg\min_{\mathbf{k}\in\mathbb{R}^T} (\|\mathbf{Ak} - \mathbf{c}\|_2^2 + \lambda^2 \|\mathbf{k}\|_2^2). \quad (3.6)$$

3.2. Полная вариация (TV)

Построим матрицу, составленную из значений $\mathbf{k}$ и $\mathbf{c}$ во всех рассматриваемых точках пространства:

$$\mathbf{K} = [\mathbf{k}_1, \mathbf{k}_2, \dots, \mathbf{k}_N], \quad \mathbf{C} = [\mathbf{c}_1, \mathbf{c}_2, \dots, \mathbf{c}_N].$$

Рассмотрим регуляризирующий метод в общем виде, как минимизацию функционала:

$$J(\mathbf{K}) = F(\mathbf{K}, \mathbf{C}) + R(\mathbf{K}, \lambda), \quad (3.7)$$

где $F(\mathbf{K}, \mathbf{C})$ задает некоторый функционал ошибки описания данных, $R(\mathbf{K}, \lambda)$ – регуляризирующий функционал, а λ параметр регуляризации.

В качестве функционала ошибки описания данных будем использовать норму невязки:

$$F(\mathbf{K}, \mathbf{C}) = \|\mathbf{AK} - \mathbf{C}\|_2^2. \quad (3.8)$$

В методе полной вариации в качестве регуляризирующего функционала используется функционал полной вариации. В дискретном виде он имеет вид: [7, 8]:

$$R(\mathbf{K}, \lambda) = \|\mathbf{K}\|_{TV}^\lambda = \sum_{i,j,t} \lambda_1 |\tilde{\mathbf{K}}_{i+1,j,t} - \tilde{\mathbf{K}}_{i,j,t}| + \\ + \sum_{i,j,t} \lambda_1 |\tilde{\mathbf{K}}_{i,j+1,t} - \tilde{\mathbf{K}}_{i,j,t}| + \sum_{i,j,t} \lambda_2 |\tilde{\mathbf{K}}_{i,j,t+1} - \tilde{\mathbf{K}}_{i,j,t}|, \quad (3.9)$$

где $\tilde{\mathbf{K}} \in \mathbb{R}^{N_1 \times N_2 \times T}$ представляет собой трехмерный тензор, составленный из значений $\mathbf{K}$, снова проинтерпретированных как трехмерный набор данных, а $\lambda = (\lambda_1, \lambda_2)$ – параметр регуляризации. Для производных по пространству и по времени можно использовать разные коэффициенты.

Метод полной вариации может приводить к различным артефактам, поскольку стремится сделать решение кусочно-постоянным.

3.3. Обобщенная полная вариация (TGV)

В отличие от полной вариации, обобщенная полная вариация использует еще и вторую производную. В этой работе используется следующая форма функционала обобщенной полной вариации [15]:

$$TGV_\lambda^2(z) = \lambda_1 \|\nabla z\|_1 + \lambda_2 \|\nabla(\nabla z)\|_1. \quad (3.10)$$

В одномерном случае аппроксимация на регулярной сетке записывается в виде:

$$TGV_\lambda^2(z) = \lambda_1 \sum_i |z_{i+1} - z_i| + \lambda_2 \sum_i |z_{i+1} - 2z_i + z_{i-1}|, \quad (3.11)$$

где $z = (z_1, z_2, \ldots, z_T)$ – сеточная функция.

В итоге регуляризирующий функционал имеет вид:

$$\begin{aligned} R(\mathbf{K}, \lambda) = \|\mathbf{K}\|_{TGV}^{\lambda} = \sum_{i,j,t} \lambda_1 |\tilde{\mathbf{K}}_{i+1,j,t} - \tilde{\mathbf{K}}_{i,j,t}| + \\ + \sum_{i,j,t} \lambda_2 |\tilde{\mathbf{K}}_{i+1,j,t} - 2\tilde{\mathbf{K}}_{i,j,t} + \tilde{\mathbf{K}}_{i-1,j,t}| \\ + \sum_{i,j,t} \lambda_1 |\tilde{\mathbf{K}}_{i,j+1,t} - \tilde{\mathbf{K}}_{i,j,t}| \\ + \sum_{i,j,t} \lambda_2 |\tilde{\mathbf{K}}_{i,j+1,t} - 2\tilde{\mathbf{K}}_{i,j,t} + \tilde{\mathbf{K}}_{i,j-1,t}| \\ + \sum_{i,j,t} \lambda_3 |\tilde{\mathbf{K}}_{i,j,t+1} - \tilde{\mathbf{K}}_{i,j,t}| \\ + \sum_{i,j,t} \lambda_4 |\tilde{\mathbf{K}}_{i,j,t+1} - 2\tilde{\mathbf{K}}_{i,j,t} + \tilde{\mathbf{K}}_{i,j,t-1}|. \end{aligned} \quad (3.12)$$

Здесь $\lambda = (\lambda_1, \lambda_2, \lambda_3, \lambda_4)$ – параметр регуляризации.

4. МЕТОД ОПТИМИЗАЦИИ

Для минимизации функционала (3.7) использовался метод ускоренного градиентного спуска Нестерова [16]:

$$\begin{aligned} z_0 = 0, \quad y_0 = 0, \\ z_{k+1} = y_k - \alpha_k \cdot \epsilon \cdot \nabla J(y_k), \\ y_{k+1} = z_k + \beta_k (z_{k+1} - z_k), \end{aligned} \quad (4.1)$$

где $\beta_k = 1 - 3/(k+1)$, а ϵ – параметр, задающий размер шага алгоритма. В этой работе использовалось значение $\epsilon = 10^{-8}$, и было обнаружено, что при этом число итераций метода может быть взято равным 30. Параметр α_k – это множитель, который делит шаг алгоритма пополам в случае, если значение функционала не уменьшилось на данном шаге:

$$\begin{aligned} \alpha_0 = 1, \\ \alpha_{k+1} = \begin{cases} \alpha_k, & J(z_{k+1}) < J(z_k), \\ \alpha_k/2, & J(z_{k+1}) \geq J(z_k). \end{cases} \end{aligned} \quad (4.2)$$

5. МЕТОД ПРОЕКЦИЙ

Эффективным методом при решении некорректных и плохо обусловленных задач является проектирование решений на некоторое компактное множество. В данной задаче имеется априорная информация о том, что решение $k(t)$ должно быть невозрастающей неотрицательной функцией [5, 6]. Множество ограниченных монотонно убывающих неотрицательных функций является компактом [17, 18]. Соответственно, можно использовать проекции на это множество. После каждого шага итерационного метода (например, метода TGV) мы проводили проектирование на это множество получаемого решения.

В этой работе проекция вычислялась следующим образом:

$$P[k](t) = \max(\min_{1 \leq x \leq t} k(x), 0), \quad (5.1)$$

где P – оператор проекции.

В итоге алгоритм работает следующим образом:

$$\begin{aligned} z_{k+1} = P[y_k - \alpha_k \cdot \epsilon \cdot \nabla J(y_k)], \\ y_{k+1} = z_k + \beta_k (z_{k+1} - z_k). \end{aligned} \quad (5.2)$$

Функционал J был взят таким же, как и в методе TGV:

$$J(\mathbf{K}) = \|\mathbf{AK} - \mathbf{C}\|_2^2 + \|\mathbf{K}\|_{TGV}^{\lambda}. \quad (5.3)$$

6. РЕЗУЛЬТАТЫ

Истинные значения остаточных функций $k(t)$ и характеристик кровотока для клинических данных неизвестны, поэтому для тестирования мы использовали сгенерированные данные. В качестве основы для генерации данных использовались искомые характеристики, взятые из [19]. Затем были сгенерированы остаточные функции $k(t) = C \cdot e^{-(at)^2}$. Чтобы оценить устойчивость методов к шуму, к синтезированным данным был добавлен белый аддитивный гауссовский шум. Значение σ для аддитивного шума было выбрано как половина стандартного отклонения в исходных данных.

Клинические данные были собраны с компьютерного томографа *G*E Optima 64.

Остаточные функции $k(t)$ – результат применения описанных методов – изображены на рис. 2.

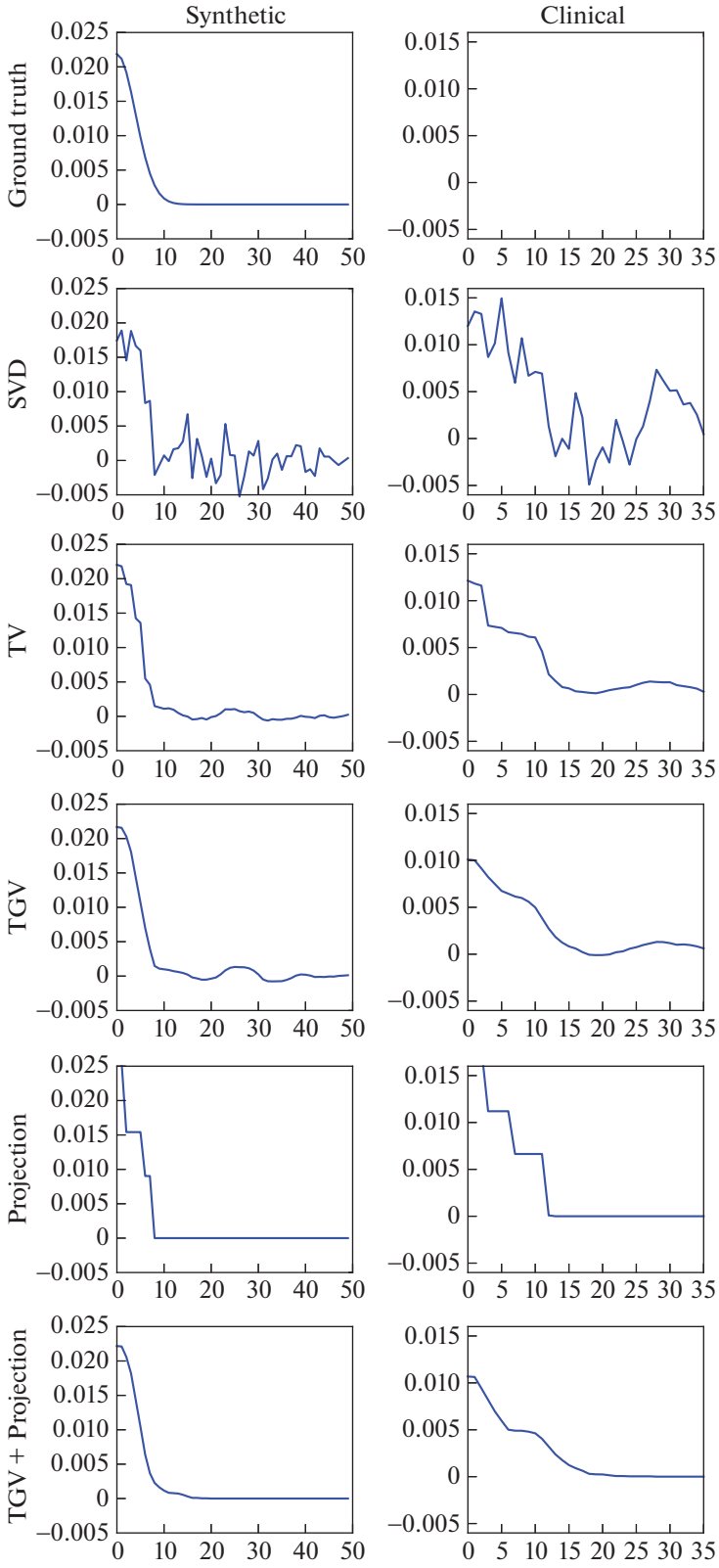


Рис. 2. $k(t)$ в одной из точек ткани мозга, полученные различными методами.

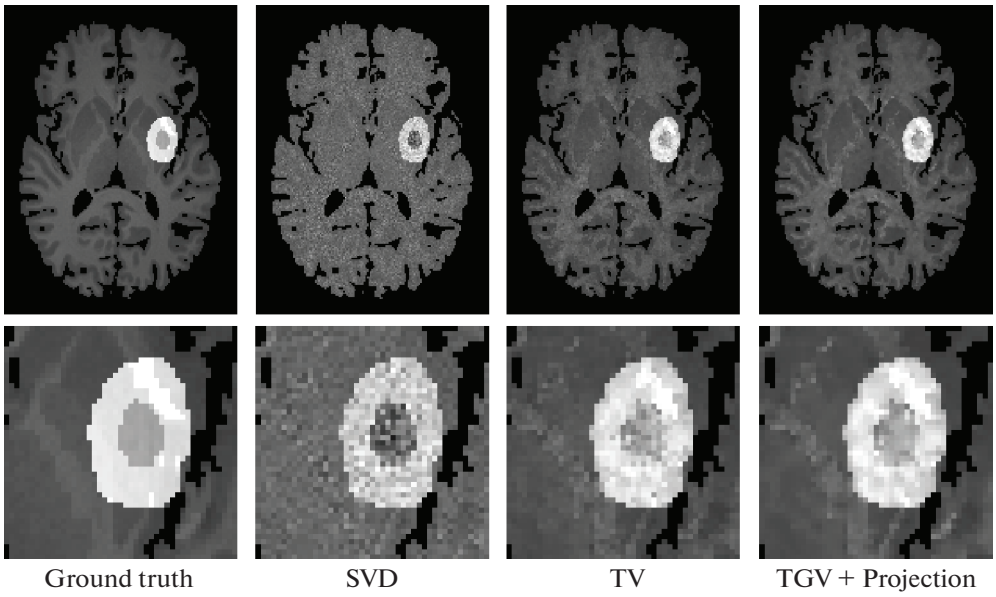


Рис. 3. Результаты расчета характеристики кровотока MTT для сгенерированных данных.

Полученные характеристики кровотока MTT искусственно сгенерированных данных изображены на рис. 3.

Можно видеть, что метод проекций, сохраняя преимущества метода TGV по сравнению с регуляризированным методом SVD, показывает больший контраст между пораженной и здоровой тканью, что повышает информативность изображения для диагностики.

СПИСОК ЛИТЕРАТУРЫ

1. *Axel Leon.* Cerebral blood flow determination by rapid-sequence computed tomography: theoretical analysis // Radiology. 1980. V. 137. № 3. P. 679–686.
2. *Корниенко В.Н., Пронин И.Н., Пьяных О.С., Фадеева Л.М.* Исследование тканевой перфузии головного мозга методом компьютерной томографии // Медицинская визуализация. 2007. № 2. С. 70–81.
3. *Сенюкова О.В., Зубов А.Ю.* Полная анатомическая разметка изображений магнитно-резонансной томографии головного мозга с помощью сопоставления с несколькими атласами // Программирование. 2016. № 6. С. 35–41.
4. *Klotz Ernst, König Matthias.* Perfusion measurements of the brain: using dynamic CT for the quantitative assessment of cerebral ischemia in acute stroke // European journal of radiology. 1999. V. 30. № 3. P. 170–184.
5. *Konstas A.A., Goldmakher G.V., Lee T.-Y., Lev M.H.* Theoretic basis and technical implementations of CT perfusion in acute ischemic stroke, part 1: theoretic basis // American Journal of Neuroradiology. 2009. V. 30. № 4. P. 662–668.
6. *Fieselmann A., Kowarschik M., Ganguly A., Hornegger J., Fahrig R.* Deconvolution-based CT and MR brain perfusion measurement: theoretical model revisited and practical implementation details // Journal of Biomedical Imaging. 2011. V. 2011. P. 14.
7. *Fang Ruogu, Sanelli P.C., Zhang Shaoting, Chen Tsuhan.* Tensor total-variation regularized deconvolution for efficient low-dose CT perfusion. // International Conference on Medical Image Computing and Computer-Assisted Intervention. 2014. P. 154–161.
8. *Fang Ruogu, Zhang Shaoting, Chen Tsuhan, Sanelli Pina C.* Robust low-dose CT perfusion deconvolution via tensor total-variation regularization. // IEEE transactions on medical imaging, 2015. V. 34. № 7. P. 1533–1548.
9. *Zhang S. et al.* High-fidelity image deconvolution for low-dose cerebral perfusion CT imaging via low-rank and total variation regularizations // Neurocomputing. 2019. V. 323. P. 175–187.
10. *Kadimesetty Venkata S., Gutta Sreedevi, Ganapathy Sriram, Yalavarthy Phaneendra K.* Convolutional Neural Network-Based Robust Denoising of Low-Dose Computed Tomography Perfusion Maps. // IEEE Transactions on Radiation and Plasma Medical Sciences. 2018. V. 3. № 2. P. 137–152.
11. *Li K., Chen G.H.* Statistical properties of cerebral CT perfusion imaging systems. Part II. Deconvolution-based systems // Medical physics. 2019. V. 46. № 11. P. 4881–4897.
12. *Zhu H. et al.* Temporally downsampled cerebral CT perfusion image restoration using deep residual learning // International journal of computer assisted radiology and surgery. 2019. P. 1–9.
13. *Robben D. et al.* Prediction of final infarct volume from native CT perfusion and treatment parameters using deep learning // Medical image analysis. 2020. V. 59. P. 101589.
14. *Golub Gene H., Reinsch Christian.* Singular value decomposition and least squares solutions. // Numerische mathematic. 1970. V. 14. № 5. P. 403–420.

15. *Nasonov A., Krylov A.* An Improvement of BM3D Image Denoising and Deblurring Algorithm by Generalized Total Variation. // 2018 7th European Workshop on Visual Information Processing (EUVIP). 2018. P. 1–4.

16. *Bottou L., Curtis F.E., Nocedal J.* Optimization methods for large-scale machine learning. // Siam Review. 2018. V. 60. № 2. P. 222–311.

17. *Тихонов А.Н., Арсенин В.Я.* Методы решения некорректных задач. Москва, Наука, 1979.

18. *Тихонов А.Н., Гончарский А.В., Степанов В.В., Ягола А.Г.* Численные методы решения некорректных задач. Москва, Наука, 1990.

19. Digital brain perfusion phantom. – https://www5.cs.fau.de/research/data/digital-brain-perfusion-phantom/.

ПРОГРАММИРОВАНИЕ, 2020, № 3, с. 42–52

КОМПЬЮТЕРНАЯ ГРАФИКА И ВИЗУАЛИЗАЦИЯ

УДК 004.92

ВИЗУАЛИЗАЦИЯ БОЛЬШИХ СЦЕН С ДЕТЕРМИНИРОВАННОЙ ДИНАМИКОЙ

© 2020 г. В. А. Семенов[a,b,c,*], В. Н. Шуткин[a,], В. А. Золотов[a,***], С. В. Морозов[a,d,****], В. И. Гонахчян[a,*****]**

[a] *Институт системного программирования им. В.П. Иванникова РАН*
109004 Москва, ул. Александра Солженицына, д. 25, Россия

[b] *Московский физико-технический институт (национальный исследовательский университет)*
141701 Московская обл., Долгопрудный, Институтский пер., д. 9, Россия

[c] *Национальный исследовательский университет "Высшая школа экономики"*
101000 Москва, Мясницкая ул., д. 20, Россия

[d] *Московский государственный университет имени М.В. Ломоносова*
119991 Москва, ул. Ленинские Горы, д. 1, Россия

**e-mail: sem@ispras.ru*

***e-mail: v451ly@ispras.ru*

****e-mail: vladislav.zolotov@ispras.ru*

*****e-mail: serg@ispras.ru*

******e-mail: pusheax@ispras.ru*

Поступила в редакцию 20.12.2019 г.
После доработки 09.01.2020 г.
Принята к публикации 19.01.2020 г.

Визуализация больших динамических сцен является актуальной проблемой компьютерной графики. Существует множество подходов к решению этой проблемы: использование отсечений областью видимости, удаление невидимых поверхностей, упрощение полигональных представлений, оптимизация техник рендеринга. Одним из эффективных методов является использование уровней детализации (LOD) для объектов сцены. Для больших сцен хорошо зарекомендовал себя иерархический метод (HLOD), в котором уровни детализации создаются для больших групп объектов. Однако данный метод сталкивается с трудностями при работе с динамическими сценами. В данной работе предлагается метод визуализации сцен с детерминированным характером динамики, основанный на использовании иерархических динамических уровней детализации (HDLOD). Описываются алгоритмы генерации кластеров HDLOD и их визуализации. Результаты проведенных вычислительных экспериментов подтверждают эффективность и перспективность предложенного метода.

DOI: 10.31857/S0132347420030073

1. ВВЕДЕНИЕ

Визуализация больших трехмерных сцен была и остается серьезной проблемой компьютерной графики. Растут мощности графического оборудования, разрабатываются новые методы и алгоритмы, однако реалистичная или достоверная визуализация сложных сцен остается недостижимой целью для многих приложений компьютерной графики. Многие индустриальные программные приложения, такие как системы CAD/CAE/CAM, BIM и GIS, являются критичными по отношению к сложности сцен и детализации индивидуальных объектов, поскольку подразумевают интерактивную модель взаимодействия человека с компьютером и возможность эффективного рендеринга сцены на оборудовании пользователя. Сегодня сцены зачастую состоят из тысяч или миллионов полигональных моделей, созданных в приложениях 3D-моделирования или полученных в результате сканирования объектов реального мира. Более того, объекты могут проявлять динамическое поведение, определяемое детерминированными или случайно происходящими событиями их появления, исчезновения или передвижения по сцене.

Существует множество подходов к решению этой проблемы: использование отсечений областью видимости, удаление невидимых поверхностей, упрощение полигональных представлений, оптимизация техник рендеринга. Одним из наиболее эффективных на сегодняшний день подходов является применение методов упрощения полигональных представлений. Все они преследуют

одну цель: сократить количество полигонов, при этом сохранив основные особенности оригинальной модели, насколько это возможно. Данные методы различаются по основной операции прореживания (удаление вершины, стягивание ребра, объединение вершин и т.д.), по метрике ошибки и по требованиям к топологии полигональной модели. Заинтересованному читателю предлагаем обратиться к [1]. Следует особенно выделить метод [2]. За счет использования стягивания ребер и квадратичной метрики ошибки этот метод показывает хорошую производительность и качество упрощения. При этом, что немаловажно, он может применяться для моделей, которые не являются односвязными двумерными многообразиями.

Одним из перспективных методов визуализации сложных сцен является использование уровней детализации (levels of detail, LOD) для объектов сцены. Это направление исследований в компьютерной графике имеет давнюю историю, начавшуюся с введения концепции LOD Джеймсом Кларком в 1976 году. Данная концепция подразумевает, что для каждого объекта сцены создается несколько версий его представления с различной степенью детализации. При визуализации сцены подходящие уровни детализации выбираются таким образом, что более точные представления используются для близких объектов, а более грубые – для дальних.

В настоящее время для визуализации больших сцен широко применяются иерархические уровни детализации (HLOD) [3, 4]. В отличие от традиционных уровней детализации, иерархические предоставляют упрощенные представления не только для индивидуальных объектов, но и для целых групп объектов, организованных в многоуровневые иерархии. Вместо выбора подходящего уровня детализации для каждого объекта становится возможным обрабатывать сразу их группы. Это позволяет достичь большей степени упрощения, сократить время обхода дерева сцены и количество вызовов отрисовки.

Однако иерархические уровни детализации затруднительно применять для произвольных динамических сцен. Каждый раз, когда объекты появляются, исчезают или двигаются, их представления должны быть пересчитаны. Время, необходимое для пересчетов, как правило, больше времени, необходимого для рендеринга сцены, что делает иерархические уровни детализации бесполезными для сцен с большим количеством динамических объектов. Известные попытки оптимизировать пересчеты благодаря использованию инкрементальных обновлений и их параллельному вычислению не привели к значимому успеху [3].

В данной статье рассматриваются динамические сцены с детерминированным характером событий, определяющих появление объектов, их исчезновение или передвижение по сцене. Под детерминированностью понимается наличие априорного знания о том, когда и с какими объектами происходят события. Для эффективной визуализации сцен такого типа предлагается новый метод, названный иерархическими динамическими уровнями детализации (HDLOD). Метод позволяет создавать иерархические уровни детализации, не требующие пересчета при анимации динамической сцены. Он принципиально отличается от методов использования уровней детализации при рендеринге сцен с типовыми геометрическими моделями и предопределенными шаблонами поведения, например, сцен моделирования пешеходных потоков [5].

Метод HDLOD не препятствует использованию разных методов рендеринга, включая методы отсечения конусом видимости (frustum culling) и методы удаления невидимых поверхностей (occlusion culling) 5. В предположении, что упрощенное представление сцены загружается в видеопамять, допустимо применение распространенных методов выполнения проверок видимости на графическом процессоре [7–9]. В статье описываются алгоритмы генерации кластеров HDLOD и их визуализации с использованием различных методов рендеринга, а также приводятся результаты проведенных вычислительных экспериментов, которые подтверждают эффективность и перспективность предложенного метода.

2. ИЕРАРХИЧЕСКИЕ ДИНАМИЧЕСКИЕ УРОВНИ ДЕТАЛИЗАЦИИ

Пусть сцена $S(t)$ определена в трехмерном евклидовом пространстве E^3 на моделируемом временном периоде $t \in [0, T]$ и представлена как линейный список объектов $s(g_s, v_s, p_s) \in S$. Каждый объект имеет неизменное геометрическое представление $g_s \subseteq E^3$. Статус присутствия объектов в сцене определяется их функциями видимости $v_s(t) : [0,T] \to \{0, 1\}$ таким образом, что функция $v_s(t)$ принимает единичное значение, если объект s присутствует в сцене в момент времени t, и нулевое значение – если отсутствует. Положение объекта определяется функцией положения $p_s(t) : [0,T] \to M$, где M – множество матриц размерности 4×4. Следует отметить, что движение объектов может быть задано множеством различных способов, например, как набор ключевых точек, в которых заданы позиция и ориентация объекта, с указанием времени, когда объект находится в этой ключевой точке. Положение объекта в каждый момент времени может определяться как результат интерполяции положений в ближайших ключевых точках. Однако, для целей рендеринга, как правило, используется задание положения объекта с помощью модельной матрицы преобразова-

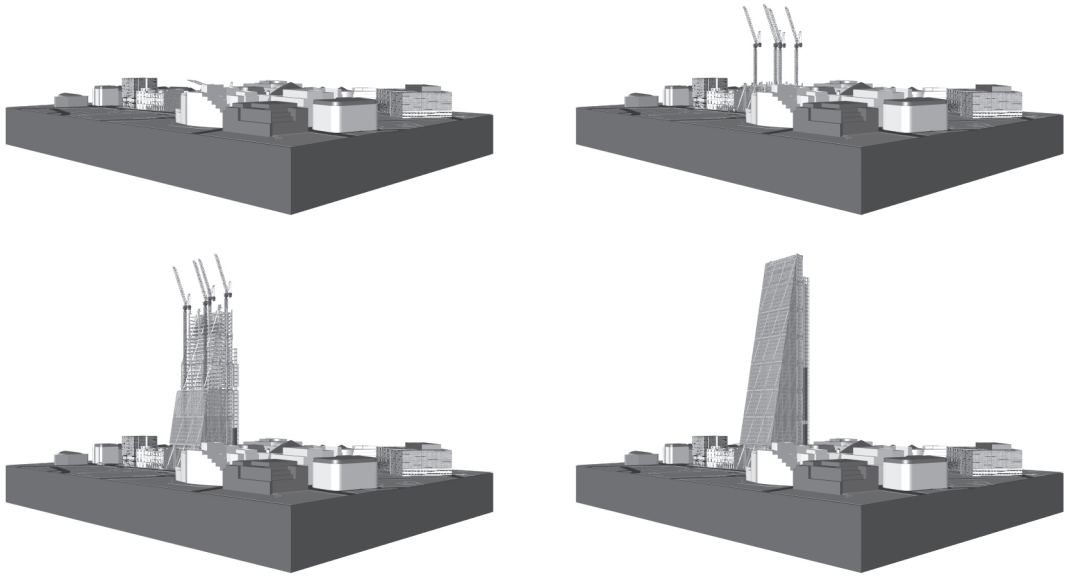


Рис. 1. Пример динамической сцены, моделирующей строительство небоскреба.

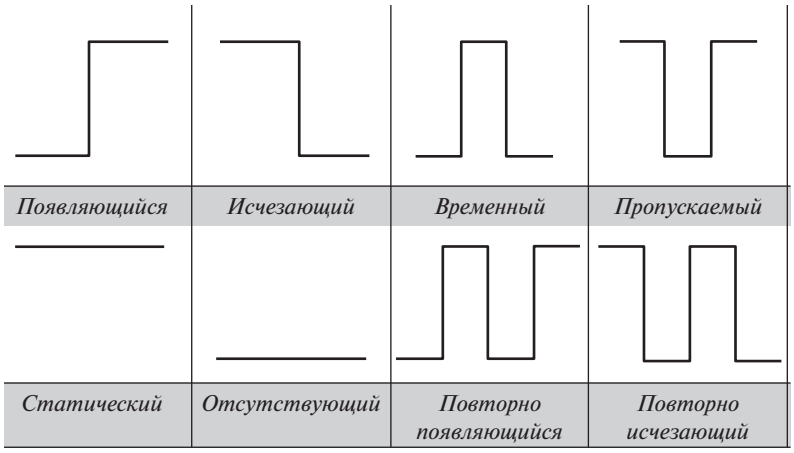


Рис. 2. Некоторые типичные функции видимости.

ния, более того, любое представление положения объекта может быть выражено при помощи матрицы. Поэтому без ограничения общности можно утверждать, что в каждый момент времени $t \in [0,T]$ положение объекта может быть представлено некоторой матрицей.

На рис. 1 показан пример динамической сцены, моделирующей строительство небоскреба в соответствии с предварительно подготовленным планом проекта. По мере изменения времени моделирования элементы конструкций и оборудование устанавливаются на строительной площадке или удаляются. Такие появления и исчезновения можно воспроизвести при помощи функций видимости, представленных на рис. 2. Элементы ландшафта остаются неизменными на протяжении всего моделируемого периода. Также в ходе строительства по строительной площадке перемещается различная техника (рис. 3).

В дальнейшем будем предполагать, что геометрическое представление объектов сцены задано набором треугольников. Не будем делать никаких предположений о топологии граничных представлений объектов и не будем требовать, чтобы граничные представления были связными или являлись многообразиями, поскольку часто для целей рендеринга предоставляется лишь так называемый “полигональный суп” без гарантий каких-либо топологических свойств.

По динамическому поведению все объекты сцены могут быть поделены на три класса.

Рис. 3. Пример динамических объектов в сцене – строительная техника и оборудование.

В класс *статических* объектов попадают объекты, видимость и положение которых не изменяется на протяжении всего моделируемого периода, то есть $\forall t \in [0,T]\, v_s(t) = 1, p_s(t) = I$, где I – единичная матрица.

Класс *псевдодинамических* объектов состоит из объектов, которые появляются и исчезают, однако их положение остается неизменным: $\forall t \in [0, T] p_s(t) = I$, но $\exists t_1, t_2 \in [0,T]$, такие что $v_s(t_1) \neq v_s(t_2)$.

Наконец, класс *динамических* объектов подразумевает, что и функция видимости, и функция положения объекта меняются с течением времени: $\exists t_1, t_2, t_3, t_4 \in [0,T]$, такие что $v_s(t_1) \neq v_s(t_2)$, $p_s(t_3) \neq p_s(t_4)$.

Будем называть иерархическими динамическими уровнями детализации (HDLOD) дерево кластеров $C(G, V, P) = \{c(g_c, v_c, p_c), \prec\}$, представленное множеством кластеров $c(g_c, v_c, p_c)$ с приписанными геометрическими представлениями g_c, функциями видимости $v_c(t) : [0,T] \to [0,1]$ и функциями положения $p_c(t) : [0,T] \to M$. В отличие от функций видимости объектов $v_s(t)$, которые принимают значения 0 или 1, функции видимости кластеров $v_c(t)$ принимают значения в диапазоне от 0 до 1, используя таким образом понятие частичной истины. Действительно, поскольку некоторые из объектов кластера могут присутствовать в сцене в некоторый момент времени, в то время как другие могут в этот же момент отсутствовать, невозможно вынести однозначный вердикт о статусе присутствия всего кластера. Для кластеров также определено отношение агломерации $\prec$ таким образом, что $c' \prec c$ тогда и только тогда, когда кластер $c' \in C$ является прямым потомком (в дереве) кластера $c \in C$.

Первым этапом генерации HDLOD является классификация объектов сцены на статические, псевдодинамические и динамические. Для каждого класса используется свой метод формирования кластеров.

Статические и псевдодинамические деревья кластеров имеют похожую структуру. В них листья дерева являются точно заданными индивидуальными объектами. Внутренние узлы представляют собой кластеры, агрегирующие геометрию (а в случае с псевдодинамическими – еще и видимость) соответствующих поддеревьев. При этом с ростом уровня в дереве геометрия и функция видимости кластеров становятся все более упрощенными, а корень представляет собой наиболее упрощенное представление огромной группы объектов.

Динамические объекты могут иметь различные траектории движения и различные моменты появления и исчезновения. Это существенно осложняет анализ и препятствует их эффективному объединению в кластеры. Поэтому для каждого динамического объекта создается отдельный кластер. В типичных сценах различные объекты могут быть представлены одной и той же моделью. Например, по строительной площадке могут перемещаться несколько экскаваторов, которые имеют одинаковое геометрическое представление, то есть являются экземплярами одной общей

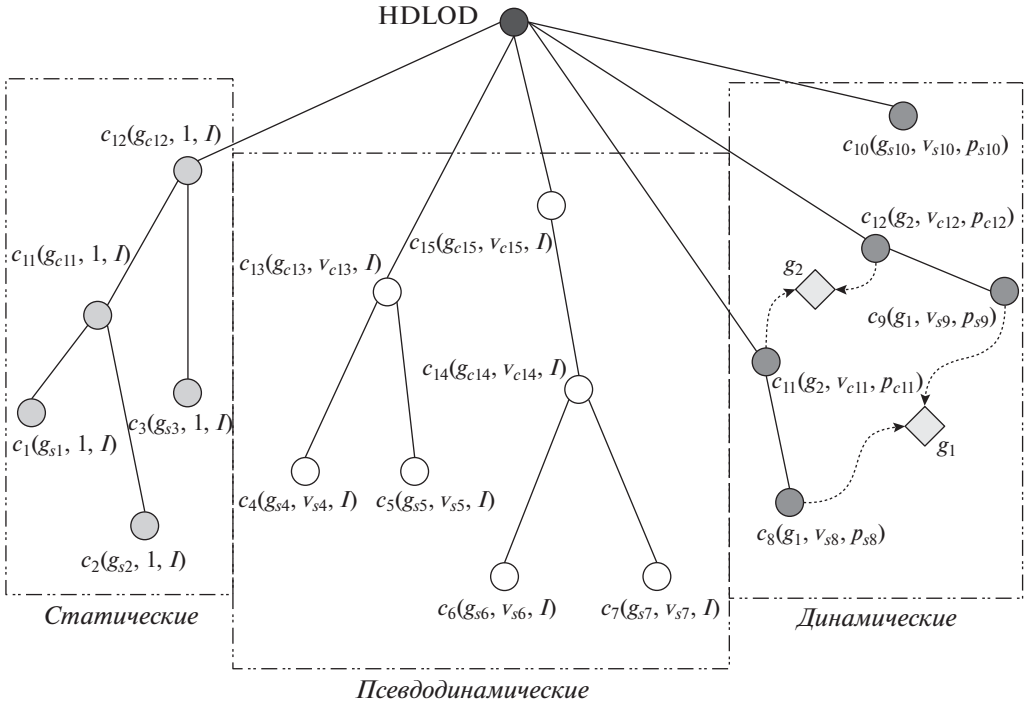


Рис. 4. Пример дерева HDLOD.

модели экскаватора. Это учитывается при формировании кластеров для динамических объектов: они могут ссылаться на общее геометрическое представление. Следует отметить, что для кластеров динамических объектов возможно дополнительно создать родительские кластеры, которые будут содержать упрощенные геометрию, функцию видимости и функцию положения.

В конечном итоге все корневые узлы деревьев статических, псевдодинамических и динамических кластеров прикрепляются к виртуальному узлу, образуя единое дерево HDLOD. На рис. 4 показан пример такого дерева.

Каждый кластер $c \in C$ хранит не только геометрическое и поведенческое представление, но также и такие производные атрибуты, как: ограничивающий параллелепипед b_c, размер или мощность w_c, а также пространственную погрешность ε_c и временную погрешность γ_c, которые будут определены ниже. Фактически эти атрибуты насчитываются при генерации иерархических динамических уровней детализации и используются при их отображении. Таким образом, каждый HDLOD-кластер представляется как кортеж $c(g_c, v_c, p_c, b_c, w_c, \varepsilon_c, \gamma_c)$.

Пространственную погрешность ε_c можно определить как абсолютную погрешность, которая устанавливает допустимое максимальное локальное отклонение геометрического представления кластера c от агрегированного представления объектов $o \prec .. \prec c' \prec c$:

$$\varepsilon_c = \max_{c' \prec c}(\varepsilon_{c'}) + D_H\left(g_c, \bigcup_{c' \prec c} g_{c'}\right).$$

Здесь погрешность ε_c определена рекуррентно с использованием метрики Хаусдорфа $D_H(A, B)$, которая представляет собой наибольшее из всех расстояний от точки из одного множества до ближайшей точки другого множества:

$$D_H(A, B) = \max\{\max_{x \in A} \min_{y \in B} D(x, y), \max_{y \in B} \min_{x \in A} D(x, y)\},$$

где A, B – замкнутые множества точек и $D(x, y)$ – функция метрики в Евклидовом пространстве.

Временная погрешность γ_c определяется для псевдодинамических кластеров как максимальное отклонение функции видимости кластера от функций видимости оригинальных объектов:

$$\gamma_c = \max_{c' \prec c}(\gamma_{c'} + D_V(v_c, v_{c'})),$$

где расстояние $D_V(v_A, v_B)$ вычисляется с использованием функциональной метрики:

$$D_V(v_A(t), v_B(t)) = \frac{1}{T}\int_0^T |v_A(t) - v_B(t)|\, dt.$$

Данные параметры используются для оценки отклонения геометрии и близости временных поведений. Пространственные погрешности вы-

числяются в процессе упрощения геометрии кластера. Временные погрешности могут быть вычислены вместе с функцией видимости кластера. Для вычисления функции видимости кластера предлагается использовать следующую формулу:

$$v_c(t) = \frac{\sum_{c' \prec c} w_{c'} v_{c'}(t)}{\sum_{c' \prec c} w_{c'}}.$$

Использование взвешенных сумм позволяет в большей степени учитывать поведение значимых объектов и надлежащим образом воспроизводить поведение кластера. Функция $v_c(t)$ выражает взвешенную долю видимых объектов кластера в момент времени t. Если она принимает единичное значение, это означает, что все объекты кластера присутствуют в сцене в момент времени t, если нулевое – отсутствуют в момент времени t.

Для каждого кластера необходимо принять решение о том, стоит ли отображать его, проигнорировать, или использовать более точные дочерние представления. Для этого вычисляется зависящая от времени пространственная погрешность $\delta_c(t)$ (пространственная ошибка, вызванная как геометрическим, так и временным упрощением):

$$\delta_c(t) = \begin{cases} 0, & \text{при} \quad v_c(t) = 0 \\ \varepsilon_c, & \text{при} \quad v_c(t) = 1 \\ \varepsilon_c + (1 - v_c(t)) \sum_{c' \prec c} w_{c'}, & \text{при} \quad \frac{1}{2} \le v_c(t) < 1 \\ \varepsilon_c + v_c(t) \sum_{c' \prec c} w_{c'}, & \text{при} \quad 0 < v_c(t) < \frac{1}{2} \end{cases}$$

При визуализации сцены совершается обход дерева кластеров. В каждом узле атрибуты кластера анализируются для определения необходимости дальнейшего обхода поддерева. Проверяется, присутствует ли кластер в сцене в указанное время моделирования ($v_c(t) > 0.5$), попадает ли ограничивающий параллелепипед b_c кластера в конус видимости, а также является ли погрешность $\delta_c(t)$ кластера достаточной для получения необходимого качества. Если все условия удовлетворены, представление кластера немедленно выбирается для отображения. В данном случае все поддерево может быть исключено из обхода. Если третье условие не удовлетворяется, продолжается обход поддерева кластеров и дочерние узлы проходят такие же проверки, пока не будут достигнуты листовые узлы с точно заданными объектами сцены. Псевдокод алгоритма визуализации HDLOD представлен в приложении 1.

3. ГЕНЕРАЦИЯ HDLOD

Иерархические динамические уровни детализации могут быть сгенерированы автоматически при помощи предложенного метода. Первым этапом генерации HDLOD является классификация объектов сцены на статические, псевдодинамические и динамические. Для каждого класса используется свой метод формирования кластеров.

Для начала рассмотрим метод обработки псевдодинамических объектов как наиболее сложный. Данный метод использует иерархическую (снизу-вверх) кластеризацию и многоуровневый контроль точности, он начинает с индивидуальных объектов и последовательно группирует их во все большие и большие кластеры, пока не будет достигнуто желаемое количество уровней. Корневые кластеры могут быть в дальнейшем еще сильнее упрощены, если потребуется отображать эту сцену как часть более сложной композиции. Кластеризация должна проводиться при строгом контроле точности, результирующее дерево кластеров должно удовлетворять множеству требований касательно ожидаемого количества уровней детализации, степени узлов, пространственной плотности и перекрытия дочерних кластеров, их временной близости и снижения сложности кластеров с повышением уровня.

К сожалению, классические методы кластеризации не могут быть напрямую применены к проблемам генерации HDLOD. Предъявляемые требования достаточно сложны и могут противоречить друг другу. Это препятствует математической формализации метрических функций и критериев связности, необходимых для методов кластеризации [10]. Неприемлемо высокая вычислительная сложность этих методов является другой причиной, препятствующей адаптации классических результатов. Например, наивная реализация агломеративной кластеризации имеет временную сложность $O(n^3)$ и требует $O(n^3)$ памяти, что делает ее неприменимой даже для достаточно простых сцен. Более быстрая иерархическая кластеризация с временной сложностью $O(n^2)$ и потреблением памяти $O(n^2)$ также не подходит для решения обсуждаемых проблем [10].

Поэтому предлагается формировать дерево, руководствуясь другими принципами. Процесс кластеризации разделяется на шаги, на каждом из которых сформированные кластеры удовлетворяют определенным требованиям по точности. По мере того как делаются новые шаги, эти требования ослабляются таким образом, чтобы гарантировать завершение процесса после определенного количества шагов, равного числу уровней детализации L.

На каждом запланированном шаге метода l ($1 \le l \le L$) делается попытка сформировать новые

кластеры с размерами $w_c \le w(l)$ и погрешностями $\varepsilon_c \le \varepsilon(l)$, $\gamma_c \le \gamma(l)$. Для этого пороговые значения $w(l)$, $\varepsilon(l)$ и $\gamma(l)$ выбираются таким образом, чтобы они монотонно увеличивались с увеличением уровня. Например, можно предложить следующие значения:

$$w(l) = \frac{l}{L}W, \quad \varepsilon(l) = 0.01\frac{l}{L}W = 0.01w(l),$$
$$\gamma(l) = 0.25\frac{l}{L}T,$$

где W – это размер всей сцены.

Метод на каждом шаге формирует список активных кластеров, которые будут участвовать в следующем шаге. Изначально в список активных кластеров помещаются все оригинальные объекты. На каждом шаге метода представители из числа активных кластеров выбираются с использованием кривых пространственного заполнения Гильберта [11]. С одной стороны, это позволяет выбирать представителей во всем объеме сцены, с другой – локализовать их в плотно заполненных областях. Далее для каждого представителя осуществляется поиск соседей. Соседи должны удовлетворять условиям по пространственной и временной близости, а также гарантировать формирование родительского кластера с запланированной точностью. Из представителя и его соседей формируется новый кластер, а они становятся его детьми. Геометрическое представление нового кластера получается путем объединения представлений детей. Оно должно быть упрощено для достижения необходимой погрешности $\varepsilon_c \le \varepsilon(l)$, например, с использованием упомянутого алгоритма [2]. Функция видимости нового кластера определяется путем вычисления взвешенной функции видимости $v_c(t)$, приведенной выше. Новый кластер добавляется в список активных, а его дети – удаляются оттуда. Если же для текущего представителя не удалось найти подходящих соседей, он исключается из анализа на текущем шаге, но будет участвовать в следующих. Псевдокод описанного алгоритма представлен в приложении 2.

С использованием пространственной индексации кластеризация может быть осуществлена за $O(n\log n)$, где n – количество объектов сцены. В сравнении с классическими методами кластеризации, упомянутыми выше, этот результат является гораздо более приемлемым для больших сцен. Для быстрого поиска соседей могут применяться различные индексные структуры [11], в частности, для псевдодинамических сцен показывают хорошую производительность регулярные динамические окто-деревья [12]. В данной реализации использовались упорядочения по каждой из координат.

Для статических объектов применяется этот же метод кластеризации, однако для них учитывается лишь пространственный аспект. В данном случае статические объекты можно рассматривать как частный случай псевдодинамических.

Для каждого динамического объекта создается отдельный кластер. При этом осуществляется анализ геометрических представлений объектов. Если обнаружены объекты, имеющие одинаковое представление, то представление их кластеров задается при помощи ссылки на данное представление. Таким образом удается избежать дублирования данных.

4. РЕНДЕРИНГ

Эффективность визуализации HDLOD кластеров зависит от используемых методов рендеринга на графическом оборудовании. В данной работе предполагается применение двух параллельно работающих подсистем, первая из которых определяет видимые кластеры (visible surface determination), а вторая выполняет загрузку полигональных представлений кластеров в видеопамять и отправку команд на графический процессор, используя программный интерфейс OpenGL.

При визуализации HDLOD дерева осуществляется его обход с проверками видимости кластера на текущее модельное время с заданного положения камеры с учетом точности его полигонального представления. Для удовлетворяющих проверкам кластеров сообщения об их отображении передаются второй подсистеме. Получив сообщение, вторая подсистема отправляет триангулированное представление кластера и команду его рендеринга на графический процессор. После того, как все необходимые данные буферизованы в памяти графического процессора, производится вызов отрисовки (draw call), который запускает выполнение команд рендеринга. Выполнение каждой команды состоит из следующих этапов: преобразование вершин (вершинный шейдер), растеризация треугольников, расчет цвета фрагментов (фрагментный шейдер), композиция и вывод на экран (см. рис. 5).

Применение метода HDLOD позволяет уменьшить время, необходимое для выполнения всего описанного процесса. Во-первых, уменьшается количество треугольников в геометрических представлениях кластеров, что в свою очередь ускоряет передачу данных в видеопамять и сокращает время этапа преобразования вершин. Во-вторых, уменьшается общее количество команд рендеринга, что приводит к сокращению накладных расходов на их обработку.

При обработке большого количества команд расходуются ресурсы центрального процессора для валидации и записи текущего состояния графического

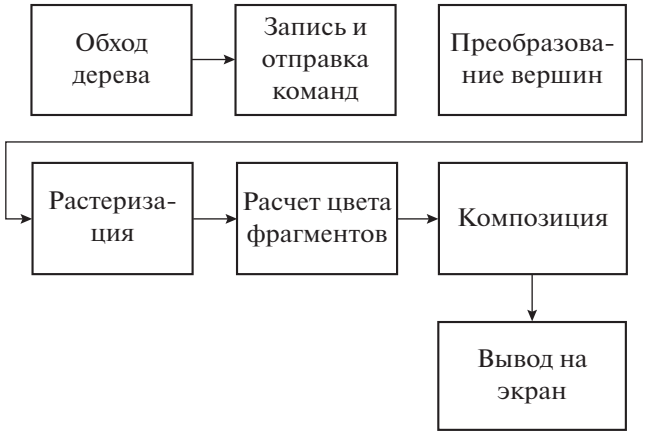


Рис. 5. Основные этапы рендеринга объектов сцены.

конвейера. Для снижения расходов можно использовать расширение OpenGL NV_Command_List, которое позволяет предварительно сформировать массив команд рендеринга вместе с текущим состоянием конвейера [13]. Это расширение является достаточно эффективным, но имеет ограниченную поддержку на современном графическом оборудовании.

В программной реализации метода визуализации HDLOD дерева использовалась процедура OpenGL MultiDrawElementsIndirect, которая позволяет выполнить массив буферизованных команд с помощью одного вызова [14]. Однако для этого требуется составить массивы команд, матриц преобразований и материалов. Для сокращения затрат на буферизацию и удаления невидимых поверхностей применяется метод пространственной декомпозиции на основе окто-деревьев (single reference octree) [15]. С каждым октантом ассоциированы соответствующие массивы, которые поддерживаются в согласованном с HDLOD деревом состоянии и обновляются по мере появления, удаления или передвижения кластеров. Если кластер переместился внутри одного октанта, то обновляется его матрица в соответствующем массиве. Если кластер переместился в другой октант, то происходит обновление массивов обоих октантов.

5. ВЫЧИСЛИТЕЛЬНЫЕ ЭКСПЕРИМЕНТЫ

Для того чтобы апробировать введенную концепцию HDLOD, а также предложенный метод для автоматической генерации и визуализации HDLOD, была проведена серия вычислительных экспериментов. Были измерены значения времени рендеринга кадра при навигации по заданной сцене при фиксированных моментах времени моделирования, а также среднее время рендеринга кадра для анимации на протяжении всего периода моделирования. В качестве тестовой была выбрана представленная на рис. 1 динамическая сцена строительства небоскреба. Данная сцена является примером типичного проекта в строительной индустрии: в ней имеются статическое окружение, псевдодинамическая модель строящегося небоскреба, конструктивные элементы которого устанавливаются согласно плану проекта, а также динамическое оборудование, использующееся при строительстве. Все объекты сцены представлены полигональными сетками. Характеристики сцены приведены в таблице 1.

Для оценки эффективности HDLOD осуществлялась визуализация сцены при различных положениях камеры. В первом положении камера была размещена на максимально близком расстоянии, при котором сцена была видна полностью. В других положениях камера размещалась на таких расстояниях, что сцена занимала одну четвертую и одну шестнадцатую области экрана. Были выбраны положения времени в начале, конце, а также в одной третьей и в двух третях моделируемого периода. Выбор таких положений камеры, при которых сцена полностью помещается на экране, обусловлен желанием исключить фактор отсечения конусом видимости. Вычислительные эксперименты проводились на компьютере типичной конфигурации: Intel Core i7-4790 CPU (3.6 GHz), 16 GB of RAM, GeForce GTX 750 Ti (2 GB).

В таблице 2 приведены результаты измерений производительности в ходе описанных экспериментов. В последней колонке содержатся результаты, полученные при рендеринге сцены без использования упрощенных представлений, а в первых трех столбцах – с использованием HDLOD дерева. Видно, что применение HDLOD значительно повышает производительность. По мере удаления камеры от сцены достигаемый эффект растет, поскольку для отображения выбираются все более крупные (и упрощенные) кластеры.

Таблица 1. Количество объектов и треугольников в сцене

	Статические	Псевдо-динамические	Динамические	Всего
Объекты	498	40010	15649	56157
Треугольники	66784	2879506	1313767	4260057

Таблица 2. Время рендеринга кадра (в миллисекундах) при визуализации сцены строительства небоскреба

	1/1 экрана	1/4 экрана	1/16 экрана	Без HDLOD
Начало	3.81	3.58	3.29	16.9
1/3 периода	17.47	17.31	15.59	35.2
2/3 периода	8.5	7.87	7.53	54.22
Конец	3.72	3.58	3.25	61.56
Анимация	9.59	8.51	7.94	47.43

Данная зависимость наблюдается как при навигации по статической сцене, зафиксированной в выбранные моменты времени, так и при анимации сцены. В данном эксперименте уровни детализации для динамических объектов не применялись, если же использовать упрощения и для динамических объектов, то рост производительности с ростом расстояния до сцены будет более значительным. С ростом сложности сцены к концу времени моделирования (строящийся небоскреб) можно наблюдать увеличение времени кадра при визуализации сцены без применения HDLOD. Однако можно заметить, что HDLOD показывают наихудшую производительность для положения времени в 1/3 модельного периода. Это обуславливается тем, что в данный момент времени в сцене происходит большое количество изменений и для многих кластеров функция видимости $f_c(t)$ оказывается близкой к 0.5, а их пространственная погрешность, зависящая от времени, $\delta_c(t)$ оказывается в этот момент очень большой, что вынуждает использовать дочерние представления. Тем не менее, производительность остается выше, чем без использования HDLOD.

Аналогичная серия экспериментов была проведена для других задач визуального моделирования проектов строительства, городских инфраструктурных программ, машиностроительных процессов. Ее результаты подтверждают высокую эффективность и перспективность предложенного метода.

6. ЗАКЛЮЧЕНИЕ

Таким образом, в работе предложен метод визуализации сцен с детерминированным характером динамики, основанный на использовании иерархических динамических уровней детализации (HDLOD). В отличие от распространенных методов уровней детализации (LOD) и их иерархических расширений (HLOD), предлагаемый метод применим к широкому классу больших динамических сцен. Для метода описаны алгоритмы генерации дерева HDLOD и его эффективной визуализации с заданной точностью. Результаты проведенных вычислительных экспериментов подтверждают эффективность и перспективность предложенного метода. Дальнейшая работа будет посвящена исследованию алгоритмических вариантов разработанного метода, а также его применению в новых индустриальных приложениях.

7. Приложение 1. Псевдокод алгоритма визуализации HDLOD

```
PROCEDURE DISPLAY_CLUSTER(CLUSTER n, VIEW v, TIME t, RESOLUTION r)
{
        IF (VALUE_OF(BEHAVIOR_FUNCTION(n), t) EQUAL 0)
                RETURN
        ELSE IF (IS_OUTSIDE_FRUSTUM(BOUNDING_BOX(n), v))
                RETURN
        ELSE IF (VALUE_OF(DELTA_FUNCTION(n), t) / DISTANCE(n, v) < r)
        {
                IF (VALUE_OF(BEHAVIOR_FUNCTION(n), t) >= 0.5)
                        RENDER(GEOMETRY(n), v))
                ELSE
                        RETURN
        }
        ELSE
        {
```

```
                SET_OF_CLUSTER children = CHILDREN_NODES(n)
                FOR_EACH (CLUSTER child IN children)
                        DISPLAY_CLUSTER(child, v, t, r)
        }
}
```

8. Приложение 2. Псевдокод алгоритма кластеризации псевдо-динамических объектов

```
PROCEDURE GENERATE_HDLOD(SCENE scene, INTEGER levels, HDLOD tree)
{
        SET_OF_CLUSTER active = NULL, next = NULL
        FOR_EACH (OBJECT object IN OBJECTS(scene))
        {
                CLUSTER cluster = FORM_CLUSTER(object)
                 ADD_TO(cluster, tree)
                 ADD_TO(cluster, active)
         }
        FOR_EACH (INTEGER step = 1 TO levels)
        {
                REAL epsilon, gamma, w
                 COMPUTE_LEVEL_THRESHOLDS(scene, levels, step, epsilon, gamma, w)
                WHILE (NOT_EMPTY(active))
                {
                        CLUSTER representative = SELECT_REPRESENTATIVE(active)
                        SET_OF_CLUSTER neighbors = FIND_NEIGHBORS(active,
                                representative, gamma, w)
                         IF (IS_EMPTY(neighbors))
                        {
                                ADD_TO(representative, next)
                                REMOVE_FROM(representative, active)
                        }
                        ELSE
                        {
                            SET_OF_CLUSTER children
                            ADD_TO(representative, children)
                            FOR_EACH(CLUSTER neighbor IN neighbors)
                                    ADD_TO(neighbor, children)
                           CLUSTER cluster = CREATE_CLUSTER(children)
                           SIMPLIFY (cluster, epsilon)
                           ADD_TO(cluster, tree)
                           ADD_TO(cluster, next)
                           REMOVE_FROM(representative, active)
                           FOR_EACH (CLUSTER neighbor IN neighbors)
                                    REMOVE_FROM(neighbor, active)
                        }
                 COPY(next, active)
                 EMPTY(next)
        }
}
```

СПИСОК ЛИТЕРАТУРЫ

1. *Luebke D. et al.* Level of Detail for 3D Graphics. Morgan Kaufmann Publishers Inc. San Francisco. 2003. 432 p.
2. *Garland M., Heckbert P.S.* Surface Simplification Using Quadric Error Metrics // SIGGRAPH '97 Proceedings of the 24th annual conference on Computer graphics and interactive techniques. 1997. P. 209–216.
3. *Erikson C. et al.* HLODs for Faster Display of Large Static and Dynamic Environments // I3D '01 Proceedings of the 2001 symposium on Interactive 3D graphics. 2001. P. 111–120.
4. *Lilley S., Cozzi P.* Cesium 3D Tiles. Beyond 2D Tiling. FOSS4G-NA Presentation, 2016, https://cesium.com/presentations/files/FOSS4GNA2016/3DTiles.pdf.
5. *Toledo L. et al.* Hierarchical Level of Detail for Varied Animated Crowds // The Visual Computer. 2014. V. 30. № 6–8. P. 949–961.
6. *Cohen-Or D., Chrysanthou Y.L., Silva C.T., Durand F.* A Survey of Visibility for Walkthrough Applications // IEEE Transactions on Visualization and Computer Graphics. 2003. V. 9. № 3. P. 412–431.
7. *Bittner J. et al.* Coherent Hierarchical Culling: Hardware Occlusion Queries Made Useful // Computer Graphics Forum. 2004. V. 23. № 3. P. 615–624.
8. *Guthe M. et al.* Near Optimal Hierarchical Culling: Performance Driven Use of Hardware Occlusion Queries // Eurographics Symposium on Rendering. 2006. P. 207–214.
9. *Mattausch O. et al.* CHC++: Coherent Hierarchical Culling Revisited // Computer Graphics Forum. 2008. V. 27. P. 221–230.
10. *Xu D., Tian Y.* A Comprehensive Survey of Clustering Algorithms // Annals of Data Science. 2015. V. 2. P. 165–193.
11. *Samet H.* Foundations of Multidimensional and Metric Data Structures. Morgan Kaufmann Publishers Inc. San Francisco, 2006. 1024 p.
12. *Morozov S. et al.* Indexing of Hierarchically Organized Spatial-Temporal Data Using Dynamic Regular Octrees // Perspectives of System Informatics, Lecture Notes in Computer Science. 2018. V. 10742. P. 276–290.
13. *Lorach T.* Approaching Zero Driver Overhead. SIGGRAPH 2014 Presentation, 2014, https://on-demand.gputechconf.com/siggraph/2015/presentation/SIG1512-Tristan-Lorach.pdf
14. *Bennett J., Carter M.* Performance Gains Achieved Through Modern OpenGL in the Siemens DirectModel Rendering Engine. GTC Presentation, 2015, http://on-demand.gputechconf.com/gtc/2015/presentation/S5387-Jeremy-Bennett.pdf
15. *Gonakhchyan V.* Efficient Command Buffer Recording for Accelerated Rendering of Large 3D Scenes // Proceedings of the 12th International Conference on Computer Graphics, Visualization, Computer Vision and Image Processing, 2018. P. 397–402.

ПРОГРАММИРОВАНИЕ, 2020, № 3, с. 53–65

КОМПЬЮТЕРНАЯ ГРАФИКА И ВИЗУАЛИЗАЦИЯ

УДК 004.92

УПРОЩЕНИЕ CAD-МОДЕЛЕЙ ПУТЕМ АВТОМАТИЧЕСКОГО РАСПОЗНАВАНИЯ И ПОДАВЛЕНИЯ ЦЕПОЧЕК СКРУГЛЕНИЙ

© 2020 г. С. Е. Сляднев[a],*, В. Е. Турлапов[a],**

[a] *Нижегородский государственный университет им. Н.И. Лобачевского*
603950 Нижний Новгород, пр. Гагарина, д. 23, Россия

**E-mail: sergey.slyadnev@gmail.com*

***E-mail: vadim.turlapov@itmm.unn.com*

Поступила в редакцию 20.12.2019 г.
После доработки 09.01.2020 г.
Принята к публикации 19.01.2020 г.

Описана процедура упрощения CAD-моделей путем распознавания и подавления некоторых типов скруглений и их цепочек. Предлагаемый метод основан на эйлеровых операторах KEV, KEF и KFMV, реализованных на базе геометрического ядра с открытыми исходными кодами. Упрощение задействует два этапа, а именно, распознавание скруглений и их подавление с гарантией топологической и геометрической целостности результата. Описанный подход ориентирован на использование в автоматическом режиме, предъявляющем высокие требования к надежности алгоритма. Ключевыми свойствами разработанного подхода являются надежность, предсказуемость результата и расширяемая архитектура, допускающая добавление новых топологических случаев без изменения основной процедуры упрощения. Распознавание состоит в построении графа смежности граней с атрибутами, содержащими информацию о типах ребер, их свойствах и предполагаемых видах скруглений. На этапе подавления, алгоритм итеративно проходит граф смежности граней, формируя цепочки скруглений. Для каждой грани в цепочке осуществляется распознавание локальной топологической ситуации, определяющей способ подавления в терминах эйлеровых операторов. Алгоритм может быть расширен путем добавления дескрипторов новых топологических ситуаций. После применения эйлеровых операторов затронутые ребра перестраиваются для получения геометрически корректного граничного представления.

DOI: 10.31857/S0132347420030085

1. ВВЕДЕНИЕ

Упрощение CAD-моделей и сборок выполняется для решения разнообразных инженерных задач. Среди прочих назовем подготовку геометрии к расчетам, сжатие данных для эффективной визуализации, защиту интеллектуальной собственности при обмене информацией, повышение производительности работы в системах проектирования, устранение вторичных конструктивных элементов для обеспечения распознавания первичных и т.п. В системах параметрического проектирования, основанных на истории построения, упрощение CAD-модели может быть выполнено путем исключения соответствующего набора конструктивных элементов из дерева операций. В случае же, если история построения отсутствует или не содержит искомых конструктивных элементов, такой подход неприменим. Возникает необходимость использования операторов прямого редактирования, которые на сегодняшний день отсутствуют в открытых библиотеках геометрического моделирования.

В данной работе мы описываем оператор подавления цепочек скруглений, основанный на распознавании конструктивных элементов с последующим применением эйлеровых операторов. Вкладом настоящей работы являются: а) реализация эйлеровых операторов KEV, KEF и KFMV на основе открытого ядра геометрического моделирования; б) разработка расширяемой архитектуры распознавания и подавления цепочек скруглений; в) реализация алгоритма подавления целого ряда конструктивно-значимых типов цепочек скруглений.

Предлагаемый алгоритм допускает работу в глобальном и локальном режимах. В первом случае осуществляется инкрементное подавление, т.е. цепочки удаляются одна за другой с промежуточным перестроением вспомогательных структур данных. Во втором – локальном – режиме, распознавание цепочки начинается из грани, вы-

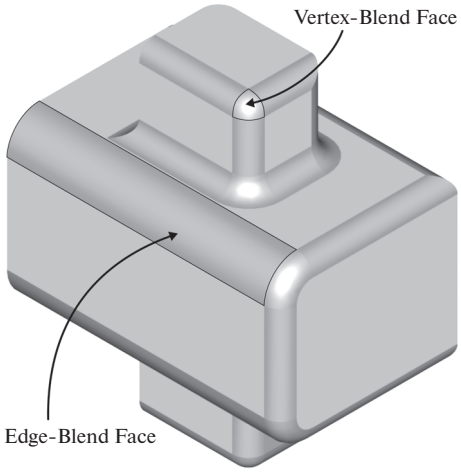


Рис. 1. Распознаваемые типы скруглений: EBF (edge-blend face) и VBF (vertex-blend-face).

бранной пользователем в качестве кандидата, притом остальные цепочки оказываются нетронуты. Алгоритм подавления накапливает непрерывную историю модификации модели, позволяющую соотнести результирующее граничное представление с исходным, обеспечивая, тем самым, возможность сохранения ассоциированных данных (цветов, имен, аннотаций, допусков и т.п.).

Далее в разделе 2 дан обзор наиболее значимых работ, посвященных распознаванию и подавлению цепочек скруглений. В разделе 3 описан наш подход к распознаванию скруглений на основе атрибутированного графа смежности граней. В разделе 4 представлен алгоритм подавления скруглений и инкрементная процедура для его применения в пакетном режиме. Раздел 5 содержит примеры, демонстрирующие использование алгоритма на CAD-моделях из повседневной практики авторов. Там же дается сравнение разработанного метода с более общим алгоритмом удаления граней. Вопросы, оставленные для дальнейшего исследования, приведены в разделе 6.

2. СОСТОЯНИЕ ПРОБЛЕМЫ

Публикации, посвященные проблеме упрощения CAD-моделей и сборок, были рассмотрены нами в работе [1]. Там же дано описание соответствующего программного комплекса. Остановимся на результатах, имеющих отношение преимущественно к подавлению цепочек скруглений.

Упрощение CAD-модели состоит в геометрическом преобразовании формы сообразно функциональным и технологическим требованиям, предъявляемым к детали. Поскольку инженерная семантика модели находит отражение в ее конструктивных элементах (КЭ), для упрощения геометрии необходимо выделить КЭ в явном виде. Наличие параметрической модели с историей построения иногда позволяет обойтись без вычислительно сложных процедур, таких как распознавание КЭ или морфологический анализ модели [2]. К сожалению, параметрические модели доступны не всегда, и даже в случае их наличия, история построения может не содержать искомых конструктивных элементов. Так, глухое отверстие, заданное в параметрической модели, может преобразоваться в сквозное при удалении из модели нового элемента объема. Аналогичным образом, скругления, заданные на плоском эскизе, при вытягивании призматического тела не будут автоматически преобразованы в конструктивные элементы скруглений на ребрах. Таким образом, для упрощения геометрии возникает необходимость в привлечении средств прямого редактирования даже в тех случаях, когда параметрическая модель доступна.

Авторы работ [3, 4] описывают процедуру подавления скруглений, реализованную в популярной САПР Rhinoceros. Алгоритм выполняет предварительную классификацию граней по типам EBF (edge-blend face) и VBF (vertex-blend face) (рис. 1). Затем алгоритм рассчитывает новые геометрические носители граничных элементов (ребро для скруглений типа EBF и вершина для VBF). Топология модели изменяется на завершающем этапе процедуры и не предугадывается изначально.

Авторы работ [5, 6] описывают “топологическую” процедуру подавления скруглений, основанную на применении эйлеровых операторов с последующим “стягиванием” геометрии. В отличие от эвристических “геометрических” подходов, таких как [4], алгоритмы [5, 6] заранее подготавливают программу преобразования топологии модели в виде последовательности эйлеровых операторов, гарантирующих целостность граничного представления. Предсказуемость результата, достижимая благодаря локальному анализу топологии скруглений, делает, на наш взгляд, алгоритмы [5, 6] предпочтительными для автоматического упрощения CAD-моделей в пакетном режиме. Так, сложные конфигурации скруглений, которые не могут быть обработаны в силу ограничений алгоритма, окажутся нераспознанными еще на начальной стадии и будут пропущены. Таким образом, алгоритм примет в обработку только те топологические случаи, для которых достоверно известна соответствующая программа эйлеровых операторов. С другой стороны, реализация подобных алгоритмов требует наличия геометрического ядра, содержащего эйлеровы операторы (например, Parasolid или ACIS). Поскольку единственное открытое геометрическое ядро OpenCascade данных операторов не предоставляет, одной из

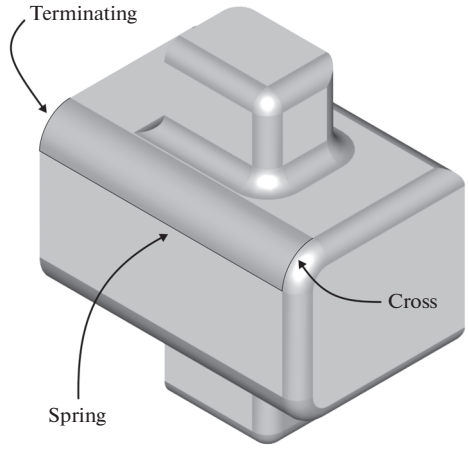


Рис. 2. Типы ребер, ограничивающих грани скруглений.

наших задач в контексте настоящего исследования являлась их реализация.

Распознавание элементов скруглений на граничном представлении считается относительно простой задачей (в отличие от задачи их подавления). Действительно, всякая грань задается точным уравнением параметрической поверхности $s(u,v)$, имеющей, как правило, явный спецификатор собственного типа на уровне структур данных (плоскость, цилиндр, сфера, тор, сплайн и т.п.). На основании данного спецификатора и дифференциальных свойств поверхности, можно заключить, является ли некоторая грань элементом скругления.

Распознавание цепочек скруглений требует локального анализа топологии в окрестности нескольких граней-кандидатов. Достаточно общий (с практической точки зрения) алгоритм распознавания цепочек, дающий также порядок их построения, был предложен в работе [7]. Работа [6] дополняет результат, оформленный в [7], алгоритмом подавления, где в качестве одного из этапов задействован оператор удаления грани [8] тех же авторов.

Комплекс работ [6–8] дает, на наш взгляд, *принципиальное* решение проблемы распознавания и подавления скруглений. Как указано выше, основная идея этого “топологического” подхода состоит в угадывании структуры результирующей модели с последующим преобразованием исходного граничного представления к заранее известному результату. Наша работа, таким образом, состоит в формализации и реализации данного принципа с использованием открытых средств геометрического моделирования.

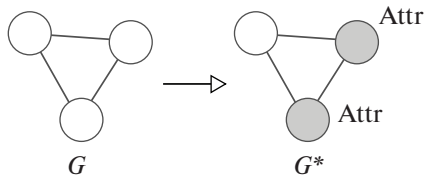


Рис. 3. Обогащение графа смежности граней G атрибутами как результат процесса распознавания скруглений.

3. РАСПОЗНАВАНИЕ СКРУГЛЕНИЙ

Для подавления цепочек скруглений их следует автоматически распознать в исходной модели. На данном этапе задача состоит не только в поиске собственно граней, но также в классификации принадлежащих им ребер (рис. 2). Классификация осуществляется по следующим типам: а) опорные ребра (spring edges); б) поперечные ребра (cross edges); в) терминальные ребра (terminating edges). Результат распознавания помещается в качестве атрибутов графа смежности граней. На данном этапе цепочки не формируются. Техника распознавания, преобразующая граф смежности граней с целью определения последовательности эйлеровых операторов, ранее не публиковалась.

Кратко, процесс распознавания состоит в следующем:

1. Построение атрибутированного графа смежности граней (attributed adjacency graph, AAG).

2. Распознавание граней типа EBF (edge-blend faces).

3. Распознавание граней типа VBF (vertex-blend faces).

4. Уточнение терминальных ребер (терминальное ребро не может содержаться в двух гранях, распознанных как элементы скруглений).

5. Извлечение результата.

Для распознавания граней типа EBF и VBF используется набор геометрических и топологических эвристик, рассматриваемых ниже.

В результате процедуры распознавания, исходный граф смежности граней G преобразуется в эквивалентный граф G^*, снабженный дополнительными атрибутами, ассоциированными с его вершинами (рис. 3). Основными являются два типа атрибутов: а) элемент скругления (значение атрибута $A(\cdot) = 1$); б) несущая грань (значение атрибута $A(\cdot) = 2$).

3.1. Граф смежности граней

Ключевым аппаратом предлагаемой архитектуры распознавания является атрибутированный граф смежности граней (attributed adjacency graph,

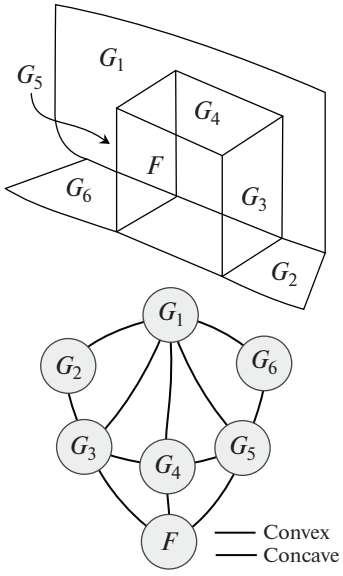


Рис. 4. Атрибутированный граф смежности граней.

AAG). Эта структура данных может применяться для решения широкого круга задач распознавания. Так, благодаря графу смежности, содержащему данные о выпуклости и вогнутости ребер (рис. 4), удается различить некоторые базовые конструктивные элементы, например, цилиндрические отверстия, карманы и выступы [14]. AAG является удобной структурой хранения данных, позволяющей акцентировать отношение смежности, извлекаемое путем анализа топологии граничного представления модели. Перечислим основные достоинства графа смежности граней в решении задач распознавания конструктивных элементов.

Во-первых, AAG позволяет осуществить переход от геометрического описания модели к соответствующему формализму теории графов. В процедурах типизации моделей (распознавание листовых тел, труб, 2.5D-моделей и проч.) применение находят такие операции на графах, как выделение компонент связности, поиск подграфа, нахождение кратчайшего пути, анализ степеней вершин и т.д.

Во-вторых, граф смежности играет роль кратковременного хранилища (кэша) для идентификаторов ребер, вершин и граней модели. Граничное представление модели в библиотеке OpenCascade не содержит уникальных идентификаторов граничных элементов в явном виде. Эти идентификаторы должны быть извлечены в процессе анализа топологических структур, что может негативно сказаться на производительности соответствующего программного обеспечения. Наличие кэша идентификаторов в составе AAG позволяет использовать его в качестве “ускоряющей структуры данных”.

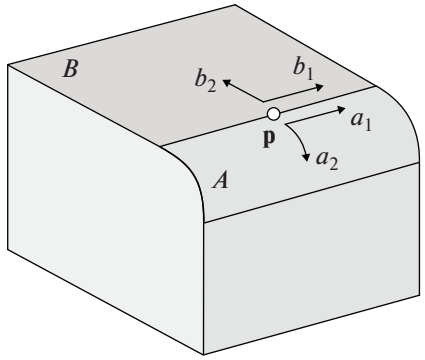


Рис. 5. Анализ кривизны для определения граней скругления.

В-третьих, AAG является носителем семантики граничного представления, дополняя его информацией об извлеченных в процессе распознавания конструктивных элементах.

3.2. Распознавание граней EBF

Распознавание граней типа EBF опирается на классификацию всех ребер модели по следующим типам (рис. 2).

1. Опорное ребро (spring edge). Опорные ребра — это гладкие ребра-направляющие для скруглений типа EBF.

2. Поперечное ребро (cross edge). Поперечные ребра могут быть гладкими и негладкими. Они характеризуются тем, что в них реализуется смежность двух граней скруглений (EBF или VBF).

3. Терминальное ребро (terminating edge). Эти ребра завершают цепочки скруглений. В случае замкнутой цепочки терминальные ребра отсутствуют.

На первом этапе выполняется поиск всех гладких ребер. Поскольку классификация двугранных углов осуществляется еще на этапе построения AAG, данный этап сводится к выборке соответствующих ребер по атрибутам из графа смежности.

Из набора гладких ребер выделяются опорные ребра граней типа EBF. С этой целью осуществляется анализ главных кривизн на ребре-кандидате. Соотношения между абсолютными величинами кривизн позволяют сделать предположение о том, является ли грань A с кривизнами a_1, a_2 гранью скругления (рис. 5). Так, должны выполняться следующие соотношения:

1. Абсолютная величина a_2 превосходит абсолютную величину a_1, то есть кривизна в поперечном направлении предполагаемого скругления доминирует.

2. Абсолютная величина a_2 превосходит абсолютную величину b_2 не менее, чем в c раз, где $c > 1$.

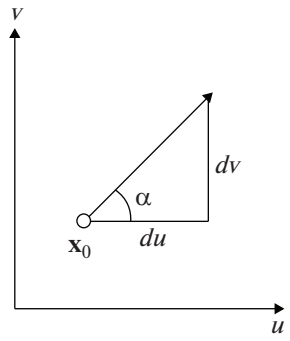


Рис. 6. К измерению кривизны вдоль ребра.

Эта эвристика выражает то наблюдение, что искомая грань EBF опирается на менее искривленную поверхность, например, на плоскость.

Радиус предполагаемого скругления полагается равным $1/a_2$.

Поперечные ребра можно найти из условия, что кривизна грани EBF вдоль такого ребра совпадает с главной кривизной a_2, полученной ранее. Вычисление кривизны поверхности вдоль кривой осуществляется по известной формуле дифференциальной геометрии, привлекающей коэффициенты первой и второй квадратичных форм поверхности:

$$k(\lambda) = \frac{L + 2M\lambda + N\lambda^2}{E + 2F\lambda + G\lambda^2}$$

Здесь $\lambda = dv/du = \tan\alpha$, (u,v) – параметры несущей поверхности (рис. 6).

Терминальное ребро не является гладким и не содержится во множествах опорных и поперечных ребер. Замкнутая цепочка не имеет терминальных ребер. Незамкнутая цепочка ограничена терминальными ребрами. Некоторые терминальные ребра доопределяются до негладких поперечных (см. раздел 3.4).

3.3. Распознавание граней VBF

Следующей стадией распознавания цепочек является выделение граней типа VBF (vertex-blend face). На данном этапе привлекается набор топологических эвристик, подсказанных типичными конфигурациями скруглений на реальных моделях. Во-первых, грань VBF не может соседствовать с другой гранью того же типа (эта эвристика отражает ограничение алгоритма). Во-вторых, для грани типа VBF должно найтись не менее трех смежных граней типа EBF, где смежность реализуется через поперечные (нетерминальные) ребра. Распознанные грани типа VBF получают соответствующий атрибут в графе смежности.

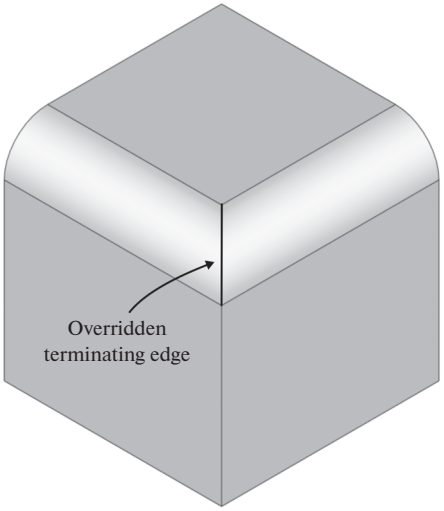


Рис. 7. Типичный пример терминального ребра, доопределенного до поперечного ребра.

3.4. Уточнение терминальных ребер

Наконец, когда все грани скруглений промаркированы, происходит уточнение терминальных ребер. Для каждого терминального ребра *e* разыскивается пара граней с индексами (f_e, g_e) таких, что $A(f_e) = A(g_e) = 1$, то есть обе грани являются элементами цепочки скруглений. Поскольку терминальное ребро не должно встречаться внутри цепочки, его тип доопределяется до поперечного ребра. Особенностью доопределенных поперечных ребер является тот факт, что они не реализуют гладких сопряжений между элементами скруглений (рис. 7).

4. ПОДАВЛЕНИЕ СКРУГЛЕНИЙ

Подавление скруглений и их цепочек заключается в удалении соответствующих граней и стягивании соседних для получения “водонепроницаемой” оболочки. Собственно удаление может быть реализовано несколькими способами: а) исключением граней из топологической структуры модели с последующим затягиванием разрывов; б) применением последовательности эйлеровых операторов. В первом случае удаление выполняется в известной степени “вслепую”, так как нет гарантии, что топологическая конфигурация в окрестности цепочки допускает корректное стягивание разрыва. Так, для управления процессом стягивания, авторы работы [10] осуществляют дополнительный анализ, сопоставляя локальной топологической конфигурации цепочки гомеоморфную ей абстрактную область. В случае, если для удаления используются эйлеровы операторы [11], топологические разрывы оказываются невозможны по определению. С другой стороны,

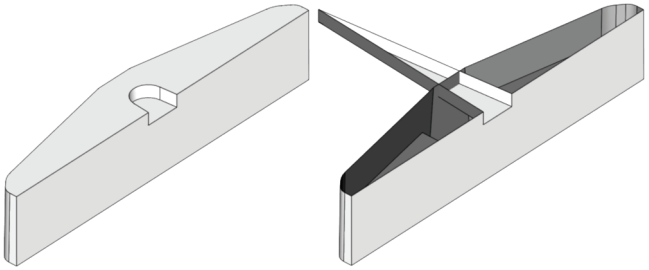


Рис. 8. Нарушение топологической целостности модели при удалении внутреннего скругления.

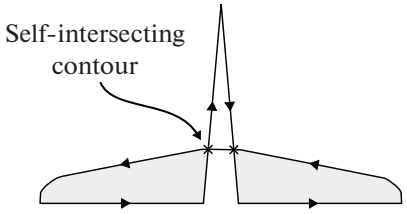


Рис. 9. Самопересечения контура базовой грани.

эйлеров оператор выполняет лишь синтаксическое преобразование модели, оставляя ее семантику (геометрию) нетронутой. Следовательно, после применения эйлеровых операторов необходимо локальное перестроение геометрии. Данный этап мы называем процессом “нормализации геометрии”.

Формально, процесс удаления скруглений состоит в следующей последовательности действий:

1. Обходом графа G^* формируются цепочки скруглений в виде множеств граней модели $F_k = \{f_j : j = N_k, M_k\}$. Здесь $k = 1, K$, где K – количество цепочек; N_k, M_k – начальный и конечный индексы граней в цепочке с номером k. Извлекаются только те вершины графа AAG, которым отвечает подграф $C_k \subset G^*$, где $A(V_i(C_k)) = 1$. Функция $V_i(\cdot)$ возвращает вершину графа f_i с номером $i = N_k, M_k$. Функция $A(\cdot)$ возвращает значение атрибута для данной вершины (1 для элементов скруглений, 2 для опорных граней и 0 для остальных).

2. Грани цепочки F_k удаляются из модели. Граф G^* играет вспомогательную роль в процессе удаления граней, поэтому его перестроение (или модификация) необходимо лишь в том случае, если подавление будет продолжено для следующей цепочки, т.е. $k \neq K$.

3. Затронутые ребра модели перестраиваются для “стягивания” опорных граней.

4.1. Базовый алгоритм

В базовом алгоритме подавления рассматриваются следующие топологические случаи для единичной грани:

1. Данная грань является изолированным скруглением.

2. Данная грань входит в замкнутую или незамкнутую цепочку скруглений, состоящую из граней типа EBF и VBF.

На предварительном этапе выполняется проверка возможности подавления цепочки в целом. С этой целью используются дополнительные топологические эвристики. В частности, любое поперечное ребро цепочки должно содержаться ровно в двух гранях той же цепочки. В противном случае цепочка оказывается распознанной неполностью и считается неподавляемой.

Последующие этапы алгоритма подавления состоят в следующем:

1. Применение операторов Эйлера для топологической подготовки результата.

2. Выполнение операции “стягивания” ребер для адаптации геометрии к новой топологической структуре (нормализация).

3. Апостериорная проверка корректности затронутых граней.

Шаг 1 состоит в последовательном применении операторов KEV (kill-edge-vertex), KEF (kill-edge-face), KFMV (kill-face-make-vertex) в порядке, определяемом типом той или иной топологической конфигурации.

Шаг 2 выполняет чисто геометрические построения на предопределенной топологической структуре модели. На данном этапе реконструируются затронутые топологической операцией ребра, вершины и грани. Так, для получения новой несущей кривой ребра, осуществляется пересечение соответствующих опорных поверхностей.

Шаг 3 предохраняет алгоритм от возможных ошибок геометрических построений (пересечение поверхностей, проецирование кривой на поверхность и т.п.), связанных с небезупречной надежностью геометрического ядра. Кроме того, на данном этапе выполняется проверка самопересечений границы области определения D для каждой затронутой грани. Последнее необходимо, поскольку подавление скруглений может, вообще говоря, привести к изменению количества компонент связности области D (рис. 8–9). Предполагается, что количество компонент связности данной области есть инвариант оператора подавления скруглений. Рассмотрение альтернативных инвариантов, таких как ограничивающий прямоугольник области D, позволит ввести новые режимы работы алгоритма подавления.

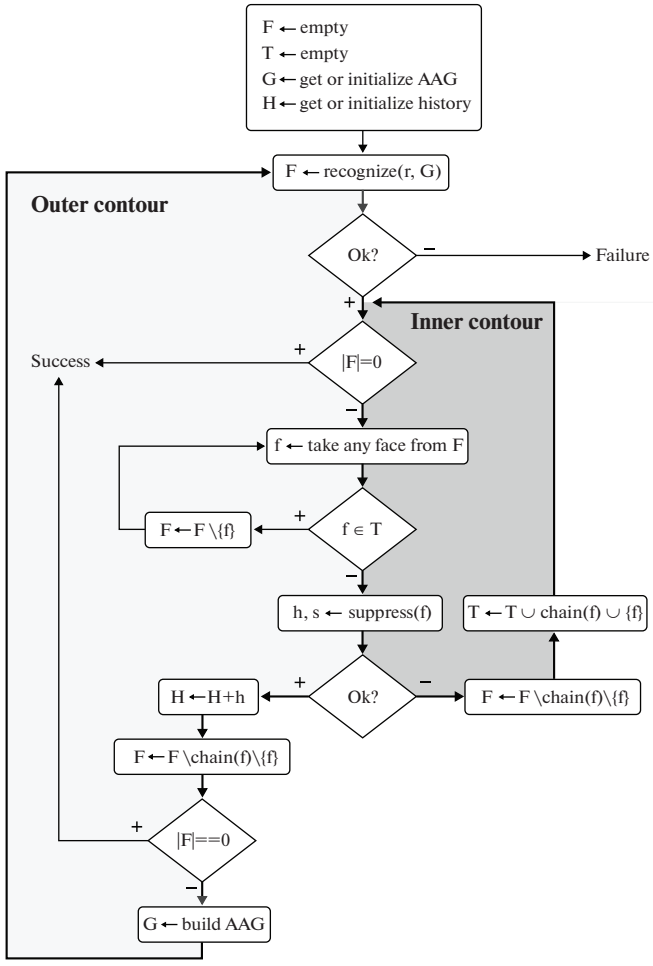


Рис. 10. Блок-схема инкрементной процедуры подавления.

4.2. Инкрементная процедура

Базовый алгоритм, описанный выше, позволяет упростить CAD-модель путем подавления единственной выбранной цепочки скруглений. Для упрощения моделей в пакетном режиме (без привлечения пользователя) этого недостаточно. Для удаления всех цепочек скруглений, реализована вспомогательная процедура так называемого инкрементного подавления.

Инкрементная процедура начинается с конструирования атрибутированного графа смежности граней *G*. Граф *G* должен быть перестроен всякий раз, когда изменяется топология модели, поскольку вершины графа адресуют граничные элементы модели через их целочисленные индексы, неустойчивые к операторам модификации [12].

Процедура реализует два вложенных цикла обработки данных (рис. 10). На каждой итерации внешнего цикла, происходит распознавание цепочек скруглений F всей модели. После выбора произвольной грани $f \in F$, выполняется попытка подавления соответствующей ей цепочки $chain(f)$. В случае успеха, граф G перестраивается для обновленной модели s и осуществляется конкатенация локальной истории модификаций h в глобальную историю H. В случае же, если цепочка $chain(f)$ не может быть удалена (внутренний цикл), соответствующие грани изымаются из очереди F на подавление, и выбирается новая цепочка без перестроения графа смежности. Кроме того, алгоритм сохраняет множество адресов неподавляемых граней T, которые, как правило, остаются актуальными даже при изменении топологии (это т.н. “транзиентные индексы” граничных элементов модели). Во внутреннем контуре алгоритм перебирает цепочки до тех пор, пока подавление не будет осуществлено, либо не опустеет очередь *F*.

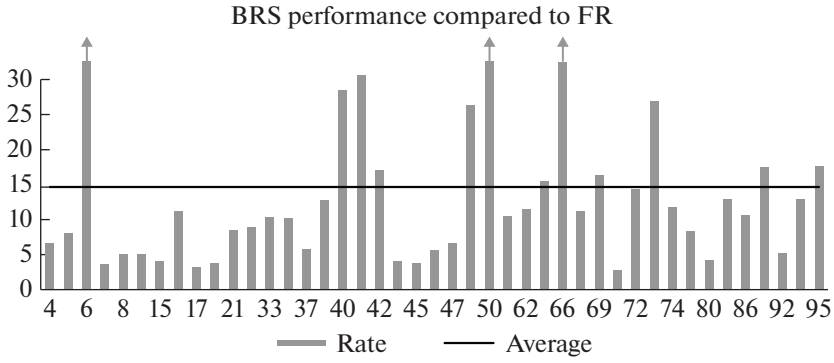


Рис. 11. Сравнение производительности алгоритма BRS с алгоритмом FR.

4.3. История модификаций

Как и любой оператор редактирования формы, оператор подавления должен сохранять ассоциативность между граничными элементами результата и исходной модели. С этой целью в алгоритм подавления введена структура данных, представляющая историю модификации топологических элементов в виде графа. Поскольку описанная выше инкрементная процедура состоит в неоднократном вызове оператора подавления, соответствующие структуры данных подлежат конкатенации (“склейке”) для получения непрерывной истории каждого граничного элемента.

5. СРАВНЕНИЕ

Для подавления скруглений и их цепочек можно задействовать оператор удаления граней общего назначения (face removal, FR). Принцип алгоритма FR состоит в изъятии целевой грани или набора граней из состава модели с последующим стягиванием соседних граней и восстановлением смежности [8]. В библиотеке OpenCascade данный подход реализован в пакете BRepAlgoAPI. Алгоритм распознавания и подавления скруглений (blends recognition and suppression, BRS), предложенный в настоящей работе, является альтернативой алгоритму FR.

В Таблице 1 приведено сравнение двух алгоритмов по трем критериям: время работы в секундах (T_{FR} и T_{BRS}), корректность полученного результата ($V_{FR}, V_{BRS} \in \{0, 1\}$), эффективность в смысле количества удаленных граней (N_{FR} и N_{BRS}). Эксперимент проводился на аппаратном обеспечении Intel(R) Core(TM) i5-9500 CPU @ 3.00GHz, 32GB RAM.

Величина r_{max} задает ограничение на максимальный радиус распознаваемых скруглений. Символом “i” (“interactive”) отмечены замеры времени для случаев, где оказалось необходимым привлечение пользователя. Алгоритмы запускались на заранее отобранных тестовых данных, доступных публично [16]. Для запуска алгоритма FR использовался предварительный этап распознавания цепочек скруглений, описанный в разделе 3. Замеры времени для алгоритма FR не включают этап распознавания и не учитывают пользовательских действий. Замеры времени для алгоритма BRS включают как распознавание, так и подавление, поскольку данные этапы в алгоритме BRS неотделимы. Символом “d” (“design intent”) отмечены результаты упрощения, являющиеся формально корректными, но нарушающие замысел проектировщика на уровне конструктивных элементов модели. Символ “?” означает, что соответствующий результат получить не удалось (алгоритм не завершился за приемлемое время, составлявшее 5–10 минут).

Ключевым свойством алгоритма автоматического упрощения CAD-моделей является его надежность. Ячейки, выделенные в Таблице 1 серым цветом, отвечают результатам, демонстрирующим недостаточную надежность соответствующего алгоритма.

Алгоритм BRS работает в среднем в 14.6 раза быстрее (рис. 11), чем алгоритм FR для случаев, когда оба алгоритма результативны, т.е. $N_{FR} \neq 0$ и $N_{BRS} \neq 0$. Более высокая производительность инкрементной процедуры BRS объясняется, прежде всего, тем, что предложенный алгоритм является локальной операцией (стр. 287 в [17], см. также [18]), тогда как задействованный алгоритм FR основан на аппарате глобальных булевых операций. Локальность алгоритма BRS гарантирует также сохранение конструктивного замысла CAD-модели, поскольку граничные элементы вне окрестности подавляемой цепочки не оказываются затронутыми.

При меньшей общей результативности ($N_{BRS} < N_{FR}$ для большинства рассмотренных случаев) алгоритм BRS оказывается более надежным, демонстрируя некорректное поведение лишь в

Таблица 1. Сравнение алгоритмов FR и BRS (инкрементной процедуры) на тестовых данных [16]. Выделены ячейки, отвечающие неудовлетворительным результатам работы того или иного алгоритма

Case	$r_{\max}$	T_{FR}	V_{FR}	N_{FR}	T_{BRS}	V_{BRS}	N_{BRS}
1	∞	0.05	1	1	0.01	1	0
2	∞	0.025	1	1	0.003	1	0
3	∞	0.07	1	2	0.008	1	0
4	∞	0.08	1	3	0.012	1	3
5	∞	0.13	1	12	0.016	1	4
6	∞	4.98	0	9	0.08	1	4
7	∞	0.018	1	1	0.005	1	1
8	∞	0.02	1	2	0.004	1	2
9	∞	0.02	1	2	0.003	1	0
10	∞	0.025	1	4	0.005	1	4
11	∞	0.035	1	7	0.006	1	0
12	∞	0.0005	1	0	0.003	1	0
13	∞	0.033	1	4	0.006	1	0
14	∞	0.01	1	4	0.007	1	0
15	∞	0.08	1	18	0.02	1	18
16	∞	0.067	1	4	0.006	1	1
17	∞	0.026	1	4	0.008	1	4
18	∞	0.075	1	5	0.02	1	5
19	∞	0.025 (i)	1	4	0.004	1	0
20	∞	0.02 (i)	1	20	0.017	1	0
21	∞	0.085	1	8	0.01	1	8
22	∞	0.045	1	2	0.005	1	2
23	∞	0.028	1	2	0.003	1	0
24	∞	0.96	1 (d)	26	0.005	1	0
25	∞	0.0004	1	0	0.004	1	0
26	∞	0.085	1 (d)	20	0.01	1	0
27	∞	0.15	1 (d)	34	0.04	1	0
28	0.45	16.7 (i)	1	44	0.64	1	0
29	∞	1.28 (i)	1	13	0.004	1	0
30	∞	0.0004	1	0	0.012	1	6
31	4	11.1	0	111	0.25	1	0
32	∞	0.0005	1	0	0.01	1	3
33	∞	2.07	1	6	0.2	1	5
34	∞	0.51	1	11	0.05	1	2
35	∞	0.0004	1	0	0.015	1	3
36	∞	44.5	1	0	8.92	1	80
37	∞	111.7	1	210	19.5	0	105
38	∞	8.14	1	47	0.64	1	27
39	∞	4.26	1	16	0.06	1	0
40	∞	13.67	1 (d)	39	0.48	1	13
41	∞	10.73	1 (d)	39	0.35	1	13

Таблица 1. Продолжение

Case	$r_{\max}$	T_{FR}	V_{FR}	N_{FR}	T_{BRS}	V_{BRS}	N_{BRS}
42	∞	1.7	1	0	0.1	1	8
43	∞	?	?	?	2.58	1	4
44	∞	0.08	1	18	0.02	1	18
45	∞	0.15	1	35	0.04	1	35
46	∞	0.017	1	1	0.003	1	1
47	∞	0.04	1	2	0.006	1	2
48	∞	11.8	1	97	0.447	1	75
49	∞	2.22 (i)	1	3	0.096	1	0
50	∞	8.11	1	42	0.19	1	8
51	∞	?	?	?	0.036	1	0
52	41	?	?	?	0.17	1	1
53	∞	3.46	1 (d)	80	0.062	1	0
54	∞	0.0005	1	0	0.01	1	3
55	∞	?	?	?	2.56	1	0
56	∞	66	1	4	0.17	1	0
57	∞	1.58	1 (d)	15	0.15	1	6
58	∞	0.5	1	11	0.003	1	0
59	∞	?	?	?	1.76	1	11
60	∞	0.0004	1	0	1.22	1	11
61	∞	?	?	?	49.62	1	90
62	∞	0.15	1	5	0.013	1	3
63	∞	0.14	1	5	0.009	1	3
64	∞	0.09	1	3	0.004	1	0
65	∞	1.01	1	0	0.07	1	8
66	∞	143.5	1	39	1.69	1	4
67	∞	1.48	1	27	0.133	1	6
68	∞	0.0004	1	0	0.14	1	12
69	∞	0.327	1	4	0.02	1	4
70	∞	11.25	1	32	3.98	1	24
71	∞	1.08 (i)	1	6	0.017	1	0
72	∞	0.591	1 (d)	12	0.041	1	4
73	∞	21.2	1 (d)	42 +	0.789	1	6
74	∞	0.82	1	4	0.07	1	4
75	∞	?	?	?	9.48	1	23
76	∞	0.15	1	4	0.018	1	4
77	∞	0.39	1	2	0.44	1	0
78	∞	?	?	?	0.21	1	0
79	5	3.1	1 (d)	32	0.022	1	0
80	∞	0.29	1	46	0.068	1	46
81	∞	28.52	1	51	0.14	1	0
82	5	?	?	?	3.39	1	21
83	∞	8.29	1	47	0.64	1	27

Таблица 1. Окончание

Case	$r_{\max}$	T_{FR}	V_{FR}	N_{FR}	T_{BRS}	V_{BRS}	N_{BRS}
84	6.5	2.08	1	30	0.071	1	0
85	∞	0.38	1	9	0.011	1	0
86	∞	0.16	1	3	0.015	1	3
87	6.5	2.14	1	34	0.004	1	0
88	∞	4.55	1 (d)	87	0.94	1	0
89	∞	9.77	1	47	0.56	1	6
90	∞	79.26	1	0	0.66	1	0
91	∞	1.25 (i)	1	0	0.053	1	0
92	∞	13.32	1	232	2.55	1	161
93	∞	0.194	1	3	0.015	1	3
94	∞	0.0004	1	0	0.01	1	3
95	∞	87.67	1	38	4.95	1	61

1.05% случаев против 23.15% для алгоритма FR. Отметим также тот неочевидный факт, что алгоритм FR не гарантирует упрощения модели, о чем свидетельствует результат на тестовом наборе #73 (см. таблицу 1). Действительно, в алгоритме FR топология модели меняется дважды: при удалении грани и при стягивании соседних граней до пересечения. Второе топологическое изменение может привести к образованию новых граней в некоторых случаях. Такие случаи возникают, как правило, при недостаточно тщательном выборе граней на удаление (рис. 12).

Некоторые результаты применения инкрементной процедуры BRS показаны на рис. 13.

6. ЗАКЛЮЧЕНИЕ

Настоящая работа заканчивает формирование нового подхода к упрощению CAD-моделей путем распознавания и подавления цепочек скруглений. В рамках данного исследования впервые реализованы эйлеровы операторы KEV, KEF, KFMV на базе открытого геометрического ядра OpenCascade и предложена архитектура распознавания конструктивных элементов на графе смежности граней G. Результатом распознавания является граф G^*, изоморфный графу G и содержащий дополнительные атрибуты, используемые на этапе подавления конструктивных элементов.

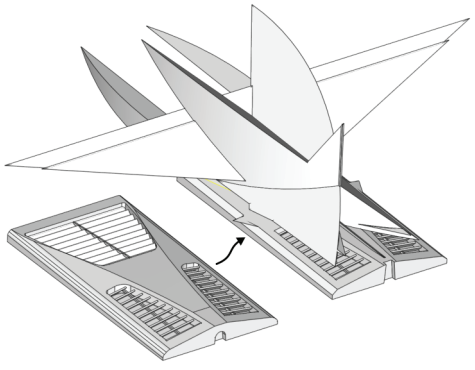


Рис. 12. Аномальный результат алгоритма FR в тестовом наборе #73 (таблица 1) при $r_{max} = \infty$.

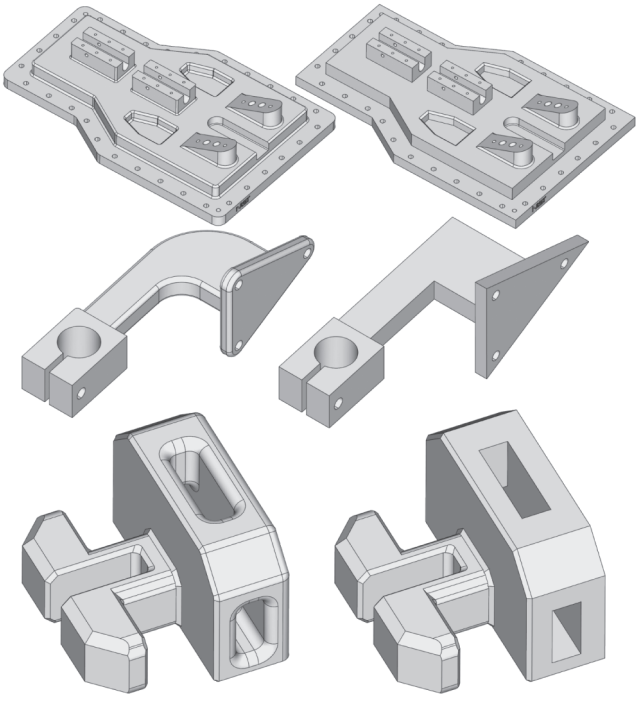


Рис. 13. Некоторые результаты применения алгоритма BRS. Сверху-вниз: тестовые наборы #92, #80, #48.

Алгоритм подавления снабжен инкрементной процедурой, позволяющей удалять обнаруженные цепочки скруглений в пакетном (batch) режиме. Алгоритм может быть расширен путем добавления дескрипторов новых топологических ситуаций. Благодаря накоплению истории модификации всех затронутых граничных элементов, конструктивно-значимая информация (цвета, имена, допуски), ассоциированная с исходной моделью, может быть перенесена на упрощенную модель без потерь.

Упрощение CAD-моделей, заданных без истории построения, остается актуальной инженерной проблемой, о чем свидетельствуют современные работы [19, 20] и другие. В рамках настоящего исследования предложена экспериментальная программная среда [13], доступная на некоммерческой основе с открытыми исходными кодами. В ее состав входят реализация атрибутированного графа смежности граней, эйлеровы операторы, алгоритм распознавания и подавления скруглений, а также соответствующая инкрементная процедура. Входными и выходными данными являются файлы формата STEP (ISO 10303). Построенный граф смежности граней может быть передан в сторонние системы, такие как MATLAB, для последующего анализа. В работе [20] передача данных AAG осуществлялась посредством формата JSON.

Предложенный подход к распознаванию и подавлению цепочек скруглений реализован на языке C++ с использованием открытого геометрического ядра OpenCascade и платформы Analysis Situs [13], где осуществлялось прототипирование. Последняя версия алгоритма и набор тестовых данных находятся в открытом доступе [9]. Алгоритм также интегрирован в коммерческое приложение CAD Processor [15].

По результатам эксперимента намечено дальнейшее развитие предложенного алгоритма. Во-первых, “каталог” обрабатываемых типов скруглений на сегодняшний день не является полным и может быть расширен. Во-вторых, представляется возможной и целесообразной оптимизация производительности инкрементной процедуры, которая выполняет глобальное перестроение графа смежности граней, несмотря на то что каждая итерация изменяет топологию локально. Перестроение графа смежности может быть заменено его модификацией с изъятием на каждой итерации граней удаленной цепочки и копированием существующих атрибутов.

Сравнение с алгоритмом FR показывает, что в некоторых случаях алгоритм BRS не справился с цепочками скруглений, на которых алгоритм FR отработал корректно. Для таких случаев алгоритм BRS должен быть доработан. Должен быть также расширен и протестирован список типов топологических конфигураций доступных алгоритму BRS.

Очевидной перспективой развития реализованного подхода является его адаптация для подавления фасок, так как это потребует лишь адаптации алгоритма распознавания, в то время как набор эйлеровых операторов остается неизменным, в силу топологической эквивалентности цепочек скруглений и фасок.

СПИСОК ЛИТЕРАТУРЫ

1. *Сляднев С.Е., Малышев А.С., Турлапов В.Е.* Автоматизированное упрощение машиностроительных CAD-моделей и сборок без использования истории построения // Труды международной конференции Графикон. 2018. С. 488–494.
2. *Belaziz M., Bouras A., Brun J.M.* Morphological analysis for product design // Computer-Aided Design. 2000. V. 32. № 5–6. P. 377–388.
3. *Lai J.Y., You Z.W., Chiu Y.K., et al.* On the Development of Blend Faces and Holes Recognition Algorithm for CAE Applications // Key Engineering Materials. 2015. № 656–657. P. 789–794.
4. *Lai J.-Y., Wong C., Huynh T.T. et al.* Small blend suppression from B-rep models in computer-aided engineering analysis // Journal of the Chinese Institute of Engineers. 2016. V. 39. № 6. P. 735–745.
5. *Cui X., Gao S., Zhou G.* An Efficient Algorithm for Recognizing and Suppressing Blend Features// Computer-Aided Design and Applications. 2004. V. 1. № 1–4. P. 421–428.
6. *Venkataraman S., Sohoni M., Rajadhyaksha R.* Removal of blends from boundary representation models // Proceedings of the seventh ACM symposium on Solid modeling and applications – SMA ’02, ACM Press. 2002. P. 83.
7. *Venkataraman S., Sohoni M., Elber G.* Blend recognition algorithm and applications // Proceedings of the sixth ACM symposium on Solid modeling and applications – SMA ’01, 2001, ACM Press. P. 99–108.
8. *Venkataraman S., Sohoni M.* Reconstruction of feature volumes and feature suppression // Proceedings of the seventh ACM symposium on Solid modeling and applications. 2002. P. 60–71.
9. Analysis Situs: suppress blend. http://analysissitus.org/features/features_suppress-blends.html.
10. *Zhu H., Menq C.* B-Rep model simplification by automatic fillet/round suppressing for efficient automatic feature recognition // Computer-Aided Design. 2002. № 34. P. 109–123.
11. *Mantyla M., Sulonen R.* GWB: A Solid Modeler with Euler Operators // IEEE Computer Graphics and Applications. 1982. V. 2. № 7. P. 17–31.
12. *Kripac J.* 1997. A mechanism for persistently naming topological entities in history-based parametric solid models // Computer-Aided Design. 1997. V. 29. P. 113–122.

13. *Slyadnev S., Malyshev A., Turlapov V.* CAD model inspection utility and prototyping framework based on OpenCascade // GraphiCon. 2017. P. 323–327.

14. *Malyshev A., Slyadnev S., Turlapov V.* Graph-based feature recognition and suppression on the solid models // GraphiCon. 2017. P. 319–322.

15. OPEN CASCADE. CAD Processor software. https://www.opencascade.com/content/cad-processor

16. Analysis Situs: test data for BRS algorithm. http://analysissitus.org/features/suppress-blends/features_suppress-blends_data.html

17. *Corney J.R., Lim T.* 3D Modelling with ACIS, Saxe-Coburg Publications; 2nd ed. Edition, 2001. 388 p.

18. *Stroud I., Nagy H.* Solid Modelling and CAD Systems: How to Survive a CAD System. Springer, 2011. 689 p.

19. *Song P.-P., Lai J.-Y., Tsai Y.-C., Hsu C.-H.* Automatic recognition and suppression of holes on mold bases for finite element applications. Engineering with Computers. 2019. V. 35. № 3. P. 925–944.

20. *Zhu F.* Application of analytical and AI-based feature detecting methods for an Energy optimized industrial process. Bachelor's thesis. 2019.

ПРОГРАММИРОВАНИЕ, 2020, № 3, с. 66–72

КОМПЬЮТЕРНАЯ ГРАФИКА И ВИЗУАЛИЗАЦИЯ

УДК 004.94

МЕТОД ИЗВЛЕЧЕНИЯ ПОВЕРХНОСТЕЙ УРОВНЯ НА GPU С ПОМОЩЬЮ ПРОГРАММИРУЕМОЙ ТЕССЕЛЯЦИИ

© 2020 г. П. Ю. Тимохин[a],*, М. В. Михайлюк[a],**

[a] *ФГУ "ФНЦ Научно-исследовательский институт системных исследований РАН"*
117218 Москва, Нахимовский пр., 36, к. 1, Россия

**E-mail: webpismo@yahoo.de*

***E-mail: mix@niisi.ras.ru*

Поступила в редакцию 20.12.2019 г.
После доработки 10.01.2020 г.
Принята к публикации 18.01.2020 г.

В статье рассматривается задача визуализации в масштабе реального времени детализированных трехмерных скалярных полей с помощью поверхностей уровня – поверхностей постоянного значения скалярного поля. Предлагается новый метод обхода ограничений, связанных с интенсивным считыванием фиктивных вершин из видеопамяти и накладными затратами на их хранение, которые возникают при реализации на GPU-методов полигонизации поверхностей уровня. Предлагаемый метод основан на эффективном создании GPU-потоков, в которых выполняется построение модели поверхности уровня, путем запрограммированной тесселяции четырехугольных патчей на регулярные сетки вершин. В работе предлагается модифицированная технология реализации на GPU "шагающих кубиков", основанная на разработанном методе, которая позволяет значительно сократить временные затраты на создание GPU-потоков и накладные затраты видеопамяти. На основе созданных решений был реализован программный комплекс построения и визуализации полигональных моделей поверхностей уровня, проведена его апробация и верификация синтезированных изображений.

DOI: 10.31857/S0132347420030103

1. ВВЕДЕНИЕ

В настоящее время во многих важных научных и прикладных областях востребована визуализация трехмерных скалярных полей, полученных в результате численного моделирования или сканирования объектов (процессов) со сложной внутренней структурой. Одним из эффективных методов является визуализация в масштабе реального времени [1] (с частотой синтеза изображений не менее 25 раз в секунду) поверхности постоянного значения скалярного поля (*поверхности уровня*). Примерами являются задачи визуализации костного скелета в медицинской компьютерной томографии [2], поверхности раздела фаз в цифровой модели керна [3], трехмерной модели рельефа с отрицательными уклонами в системах виртуального окружения [4] и др. Извлечение поверхности уровня из поля основано на вычислении в ячейках скалярной сетки точек с заданным постоянным значением поля. Чем сложнее объект исследования, тем более детальная скалярная сетка требуется для его адекватного представления, и, соответственно, тем больше вычислений необходимо произвести, чтобы извлечь поверхность уровня. В современных условиях отображения результатов визуализации на экранах сверхвысокой четкости и стереоэкранах [5] объем и время таких графических вычислений возрастают в разы, что препятствует синтезу изображений поверхности уровня в масштабе реального времени. В этой связи возникает задача создания эффективных методов и алгоритмов извлечения поверхностей уровня, основанных на распараллеливании вычислений на современных тысячеядерных графических процессорах (GPU) и использовании технологий параллельной обработки графических данных.

В данной работе предлагается новый метод решения этой задачи на GPU, основанный на полигонизации скалярного поля (извлечении набора треугольных граней, приближающего поверхность уровня), которая выполняется в масштабе реального времени в параллельных GPU-потоках, созданных с помощью программируемой тесселяции параметрических графических примитивов [6]. Предложенное решение реализовано на языке C++, языке GLSL программирования шейдеров и с использованием графической библиотеки OpenGL.

2. ПОДХОДЫ К ИЗВЛЕЧЕНИЮ ПОВЕРХНОСТЕЙ УРОВНЯ

Можно выделить два основных направления развития методов извлечения поверхностей уровня.

К первому направлению принадлежат методы, основанные на испускании лучей (“бросании лучей”, ray casting) из позиции наблюдателя через пикселы экрана до пересечения с поверхностью уровня [7–9]. Данный подход позволяет получать изображения поверхностей уровня высокого качества, однако скорость синтеза таких изображений существенно падает с ростом числа обрабатываемых пикселов (просчитываемых лучей), что препятствует визуализации в реальном времени на экранах со сверхвысоким разрешением. Последние исследования [10] показывают, что появившаяся недавно аппаратная поддержка трассировки лучей у видеокарт NVidia RTX может существенно улучшить данную ситуацию.

Ко второму направлению относятся методы, основанные на построении полигональной аппроксимации поверхности уровня в ячейках трехмерной сетки (рендер-сетки), совмещенной с сеткой скалярного поля [11–19]. При данном подходе находятся *активные* (пересекаемые поверхностью уровня) ребра и ячейки рендер-сетки, для которых по определенным правилам создаются полигоны поверхности уровня. Выделяют прямые [11–13], двойственные [14–16] и гибридные [17–19] методы полигонизации поверхности уровня. В прямых методах внутри каждой активной ячейки создаются полигоны, вершины которых лежат на активных ребрах ячейки (классическим примером является метод “шагающих кубиков” [11]). В двойственных методах полигоны создаются для активных ребер, а вершины этих полигонов создаются внутри активных ячеек. Гибридные методы являются комбинацией прямых и двойственных методов и ведут себя как двойственные методы в случаях, когда активные ячейки содержат особенности поверхности уровня, такие как острые ребра и конические вершины. Обширное исследование и сравнение методов полигонизации поверхностей уровня представлено в работе [20].

С добавлением в графический конвейер GPU программируемых (шейдерных) стадий исследования стали активно развиваться в направлении распараллеливания на GPU обработки активных ячеек и ребер [21–27]. Хорошие результаты были получены при распараллеливании прямых методов, в которых, в отличие от двойственных методов, каждая активная ячейка триангулируется независимо от других. Эффективность GPU-версий прямых методов сильно зависит от временных затрат на создание и исполнение GPU-потоков и расхода видеопамяти. В эпоху тысячеядерных GPU с иерархической структурой видеопамяти узким местом визуализации высокополигональных моделей является число обращений к глобальной видеопамяти (самой большой и медленной части видеопамяти). Снизить число таких обращений можно, выполняя построение полигональной модели для активной ячейки в геометрическом шейдере – одной из программируемых стадий графического конвейера. Чтобы запустить один GPU-поток с геометрическим шейдером, достаточно подать на его вход одну вершину. Распространенный подход состоит в создании в видеопамяти вершинного буфера (VBO) с фиктивными вершинами (по вершине на каждую ячейку рендер-сетки) и отправке этого массива вершин на графический конвейер. Фиктивные вершины поступают на вычислительные ядра GPU, где на каждом ядре параллельно, независимо друг от друга геометрический шейдер заменяет вершину на определенную полигональную конструкцию, если ячейка является активной. И хотя построение полигональной модели поверхности уровня фактически выполняется внутри GPU, в данном подходе имеются существенные ограничения, связанные с интенсивным считыванием вершинных данных из глобальной видеопамяти и дополнительными накладными затратами на их хранение. Эти ограничения препятствуют увеличению размеров рендер-сетки и построению в масштабе реального времени поверхностей уровня для сложных объектов.

В данной работе предлагается новый метод обхода описанных ограничений, основанный на задействовании в процессе построения полигональной модели поверхности уровня двух новых программируемых стадий – шейдера управления тесселяцией и шейдера вычисления тесселяции [6]. Данные стадии выполняются перед геометрическим шейдером и в общем случае предназначены для параллельной генерации на GPU большого числа связанных треугольных аппроксимаций из параметрических графических примитивов (патчей), как например, в задаче построения полигональной модели земной поверхности [28]. Данная работа основана на использовании одного из частных случаев программируемой тесселяции, при котором из четырехугольного патча (патча-квада) на GPU генерируется регулярная сетка вершин. В исследовании показано, как с помощью этой возможности можно существенно сократить время создания GPU-потоков и накладные затраты видеопамяти на примере задачи построения полигональной модели поверхности уровня с помощью “шагающих кубиков”.

3. ТЕХНОЛОГИЯ РЕАЛИЗАЦИИ НА GPU “ШАГАЮЩИХ КУБИКОВ” С ПОМОЩЬЮ ТЕССЕЛЯЦИОННЫХ ШЕЙДЕРОВ

Пусть имеется рендер-сетка R размера $m \times n \times q$ ячеек, каждому узлу (вершине) которой соответствует некоторое значение S исследуемого скалярного поля. Рассмотрим задачу построения в активных ячейках сетки R полигональной модели поверхности некоторого постоянного значения S^*. Активной будем считать каждую ячейку, для всех восьми вершин которой не выполняется условие $S < S^*$ или $S \geq S^*$, т.е. исключаются ячейки, которые лежат полностью внутри или снаружи поверхности уровня.

Рассмотрим некоторую $R_{i,\,j,\,k}$-ю ячейку. Пронумеруем вершины этой ячейки целыми числами от 0 до 7, а ребра – от 0 до 11. Запишем единичный бит для каждой вершины, у которой $S < S^*$, в противном случае – нулевой (см. рис. 1). Согласно правилу “шагающих кубиков” на ребро, вершины которого имеют разные биты (активное ребро), помещается вершина треугольника (полигона), при этом координаты P этой вершины находятся как

$$P = P_a + \frac{S^* - S_a}{S_b - S_a}(P_b - P_a), \qquad (1)$$

где P_a и P_b – координаты вершин активного ребра, а S_a и S_b – значения исследуемого скалярного поля в вершинах активного ребра. Обозначим через K конфигурацию из восьми записанных подряд полученных битов. Каждому значению K (от 0 до 255) однозначно соответствует список треугольников, который будет частью полигональной модели поверхности уровня. Каждый такой список включает в себя от 0 до 5 треугольников и задается последовательностью из 16 индексов: на каждый треугольник по три индекса активных ребер и (–1) в конце последовательности. Все 256 таких последовательностей образуют массив T списков треугольников (см. работу [12]).

Построение списка треугольников в каждой активной $R_{i,\,j,\,k}$-й ячейке будем выполнять в отдельном GPU-потоке (потоке “шагающих кубиков”). В данной работе создавать такие потоки предлагается с помощью программируемой тесселяции патчей-квадов. Запрограммировав графический конвейер определенным образом, можно параллельно разбивать каждый патч-квад на регулярную сетку размера до $D \times D$ вершин, где $D = L_{\max} + 1$, а $L_{\max}$ является *максимальным уровнем тесселяции* – наибольшим числом отрезков, на которое сторона патча-квада может быть разбита. В OpenGL версии 4.0 это число составляет не менее 64. В соответствии с архитектурой графического конвейера каждая получаемая в результате тесселяции вершина обрабатывается в отдельном GPU-потоке, который мы программируем для построения соответствующего ячейке списка треугольников (части моделируемой поверхности уровня). Важным преимуществом предлагаемого метода является то, что такие вершины (GPU-потоки) генерируется непосредственно внутри GPU, не тратя ресурс видеопамяти и не выходя за рамки графического конвейера, в отличие от решений, основанных на программно-аппаратной архитектуре CUDA.

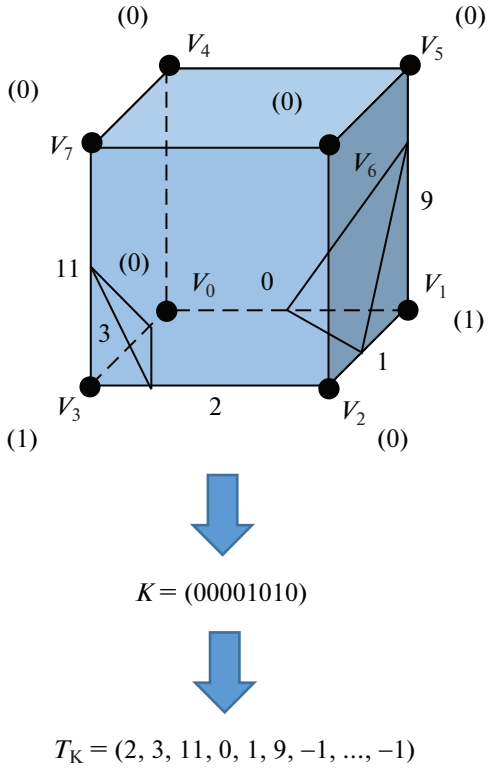


Рис. 1. Кодирование ячейки рендер-сетки.

Предлагаемая технология реализации на GPU “шагающих кубиков” включает в себя два этапа. На ***первом этапе*** (загрузка исходных данных) в видеопамять записывается вещественная 3D-текстура со значениями S вершин рендер-сетки, целочисленная 2D-текстура, в которой закодирован массив T списков треугольников, а также вершинный буфер, содержащий $m_p n_p q_p$ патчей-квадов, где $m_p = \lceil m/D \rceil$, $n_p = \lceil n/D \rceil$, $q_p = q$ (см. рис. 2). На ***втором этапе*** (визуализация) патчи-квады из вершинного буфера отправляются на графический конвейер, где распределяются между ядрами GPU и обрабатываются параллельно с помощью разработанных шейдерных программ – шейдера управления тесселяцией (TCS), шейдера вычисления тесселяции (TES), геометрического шейдера (GS) и фрагментного шейдера (FS). Далее рассмотрим их работу более подробно.

Стадия TCS-шейдера. На данной стадии выполняется вычисление двух наборов параметров.

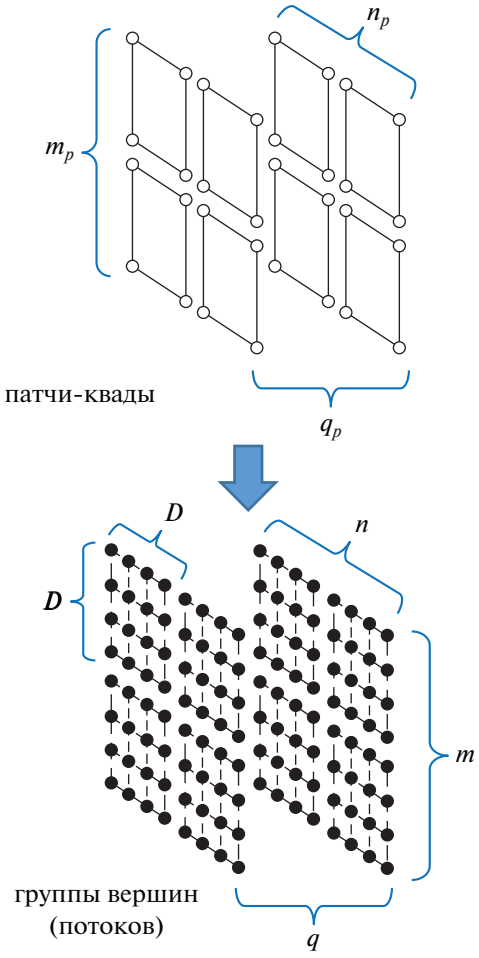


Рис. 2. Генерация потоков “шагающих кубиков”.

Первый набор является тройкой индексов (i_p, j_p, k_p) строки, столбца и слоя обрабатываемого TCS-шейдером патча в 3D массиве патчей-квадов (см. рис. 2). Эти параметры необходимы в дальнейшем для установки соответствия созданных GPU-потоков ячейкам рендер-сетки. Расчет тройки индексов (i_p, j_p, k_p) выполняется на основе встроенной переменной gl_PrimitiveID, которая является порядковым номером g патча, принимающим значения из отрезка $[0, m_p n_p q_p - 1]$:

$$k_p = \lfloor g/N \rfloor, \quad i_p = \left\lfloor \frac{g - Nk_p}{M} \right\rfloor,$$
$$j_p = g - Nk_p - Mi_p,$$

где $N = m_p n_p$, а $M = Nq_p$. Второй набор включает в себя параметры, задающие ширину и высоту 2D сетки вершин (группы потоков “шагающих кубиков”). Эти ширина и высота определяются уровнями l_w и l_h тесселяции патча-квада, которые вычисляются как

$$l_w = \max(\min(L_{\max}, n - Dj_p - 1), 1),$$
$$l_h = \max(\min(L_{\max}, m - Di_p - 1), 1).$$

Стадия TES-шейдера. Данный шейдер осуществляет параллельно обработку каждой вершины из 2D сетки вершин, полученной в результате тесселяции патча-квада, и устанавливает в ходе этой обработки соответствие вершины (потока) ячейке рендер-сетки. Это реализуется путем добавления обрабатываемой вершине тройки атрибутов (i_c, j_c, k_c) – индексов строки, столбца и слоя соответствующей ячейки в рендер-сетке, которые вычисляются как

$$i_c = Di_p + i, \quad j_c = Dj_p + j, \quad k_c = k_p,$$

где i и j являются индексами строки и столбца вершины в 2D сетке вершин, полученной после тесселяции патча:

$$i = \lfloor l_h v + \varepsilon \rfloor, \quad j = \lfloor l_w u + \varepsilon \rfloor,$$

а $(u, v) \in [0,1]$ являются нормализованными вещественными координатами обрабатываемой вершины в 2D сетке вершин (рассчитываются автоматически при тесселяции в TES-шейдере), ε – малая константа для компенсации машинной погрешности представления вещественных чисел.

Стадия GS-шейдера. Данный шейдер принимает из TES-шейдера тройку индексов (i_c, j_c, k_c) ячейки рендер-сетки и осуществляет параллельную обработку всех таких ячеек. Обработка начинается с вычисления для (i_c, j_c, k_c)-й ячейки номера конфигурации K_{i_c, j_c, k_c}:

$$K_{i_c, j_c, k_c} = \sum_{p=0}^{7} 2^p b_p,$$

где p – порядковый номер вершины в (i_c, j_c, k_c)-й ячейке (см. рис. 1), флаг b_p равен 1, если $S_p < S^*$, а в остальных случаях – 0 (S_p-значение исследуемого скалярного поля в p-й вершине ячейки). Если $K_{i_c, j_c, k_c} = 0$ или $K_{i_c, j_c, k_c} = 255$ (ячейка полностью лежит внутри или снаружи поверхности уровня), тогда GS-шейдер завершает свою работу без создания полигональной геометрии в обрабатываемой ячейке. Во всех остальных случаях: а) по номеру K_{i_c, j_c, k_c} конфигурации из текстуры T списков треугольников извлекаются списки ребер, образующих полигоны поверхности уровня внутри (i_c, j_c, k_c)-й ячейки; б) из 3D-текстуры рендер-сетки извлекаются значения S для вершин ребер и вычисляются координаты точек на этих ребрах согласно формуле (1); в) из полученных вершин выполняется построение треугольников поверхности уровня внутри (i_c, j_c, k_c)-й ячейки согласно порядку следования ребер в извлеченном списке (более подробное описание реализации эмиссии

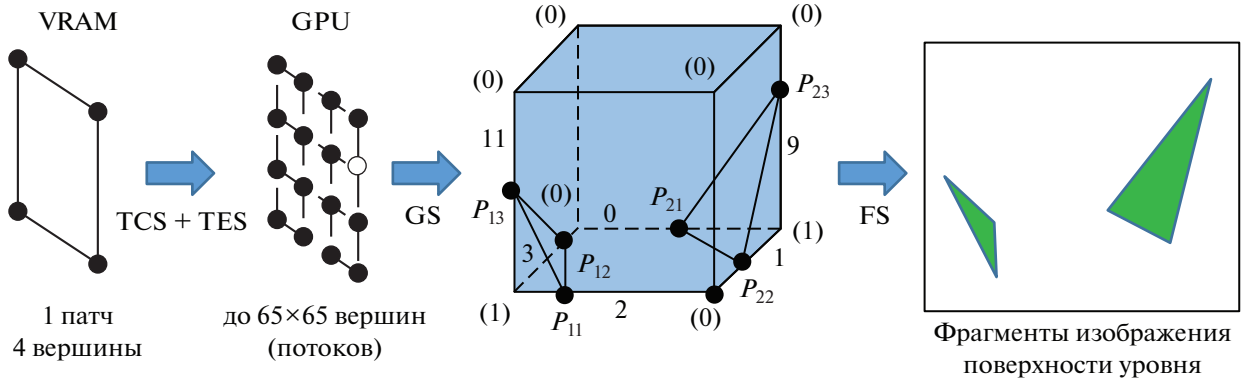


Рис. 3. Схема конвейера визуализации поверхности уровня.

треугольников в геометрическом шейдере можно найти в [29]).

Стадия FS-шейдера. Треугольники, построенные GS-шейдером проходят растеризацию (фиксированная стадия графического конвейера), в результате которой преобразуются во фрагменты изображения поверхности уровня. FS-шейдер вычисляет цвета полученных фрагментов параллельно, независимо друг от друга, на основе модели освещения Фонга [29] с направленным источником света.

На рисунке 3 показана общая схема разработанного конвейера визуализации. После прохождения всех патчей-квадов через данный конвейер в буфере кадра синтезируется результирующее изображение извлеченной поверхности уровня.

4. РЕЗУЛЬТАТЫ

На основе предложенной технологии был создан программный комплекс, реализующий извлечение полигональных моделей поверхностей уровня с помощью программируемой тесселяции. Для апробации созданного комплекса использовались трехмерные скалярные поля насыщенности воды в пористом образце нефтеносной породы, полученные в результате численного моделирования на расчетной сетке размера 100^3 ячеек. На рисунке 4 показана извлеченная поверхность, соответствующая уровню 35%-й насыщенности. Построение и визуализация полигональной модели поверхности уровня выполнялись при разрешении 3840 × 2160 с помощью видеокарты GeForce GTX 1080 Ti (3584 ядра, 11 Гб видеопамяти), средняя частота визуализации составила около 100 кадров в секунду. Полученная поверхность уровня была верифицирована путем извлечения поверхности уровня из того же массива скалярных данных с помощью системы математического и численного моделирования MATLAB [30].

Также для предложенного решения была проведена оценка накладных затрат видеопамяти (в Гб) и времени создания GPU-потоков (в миллисекундах) при увеличении размера рендер-сетки до 2000^3 ячеек. На рисунке 5 изображены графики расхода времени (T) и видеопамяти (M) для предложенной реализации (a) и для реализации распространенного подхода (b), описанного в разделе 2. По сравнению с реализацией (b) предложенное решение позволило сократить временные затраты на создание GPU-потоков в 5 раз, а накладные затраты видеопамяти – в 8 раз. На основе полученных результатов в будущем планируется провести исследование извлечения в масштабе реального времени поверхностей уровня из трехмерных скалярных полей большой протяженности (построение протяженного рельефа местности с отрицательными уклонами).

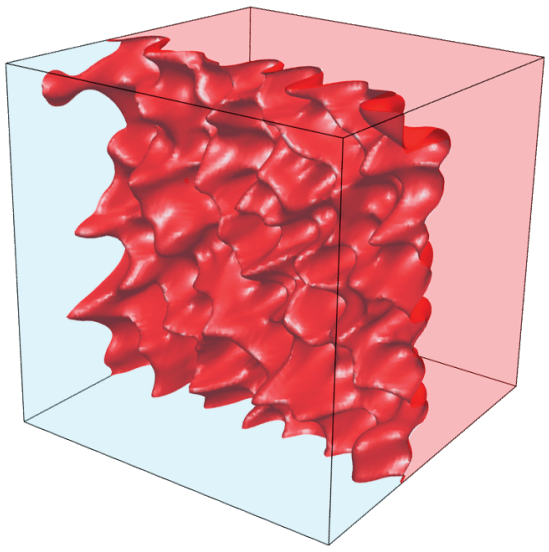


Рис. 4. Пример полученной визуализации поверхности уровня.

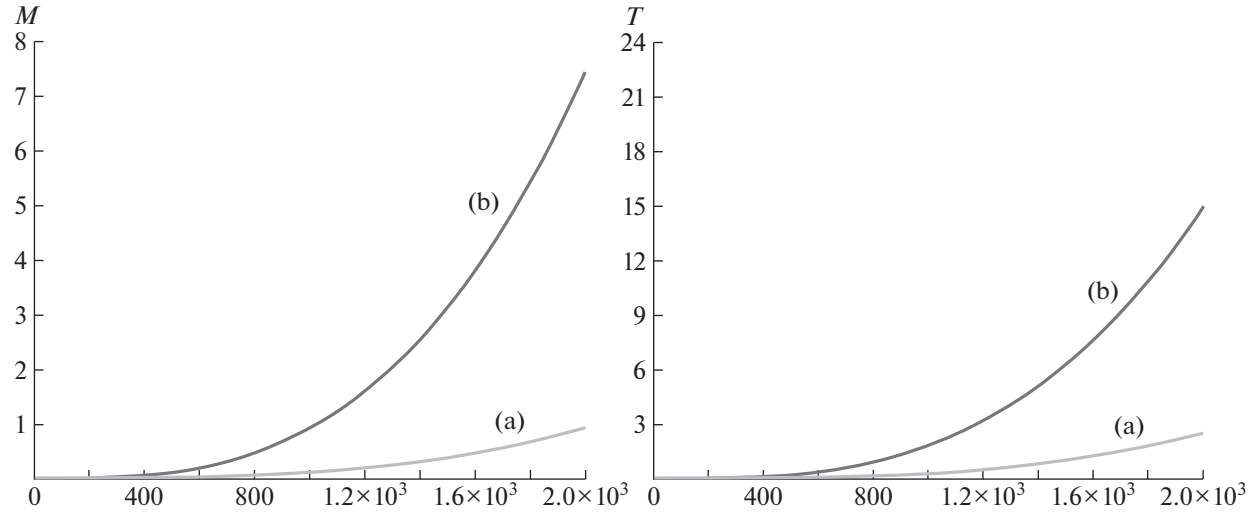


Рис. 5. Сравнение предложенного решения (a) с GPU-реализацией без тесселяционных шейдеров (b): по накладным затратам видеопамяти (слева) и по времени создания GPU-потоков (справа).

5. ЗАКЛЮЧЕНИЕ

В работе рассмотрена задача визуализации в масштабе реального времени детализированных трехмерных скалярных полей с помощью поверхностей уровня. Для преодоления выявленных ограничений, препятствующих построению в масштабе реального времени полигональных моделей поверхностей уровня сложных исследуемых объектов, предложен новый метод, основанный на выполнении на GPU-процесса извлечения поверхности уровня с помощью шейдера управления тесселяцией и шейдера вычисления тесселяции. В работе предложена эффективная технология реализации на GPU алгоритма “шагающих кубиков”, основанная на разработанном методе, которая позволяет значительно сократить временные затраты на создание GPU-потоков и накладные затраты видеопамяти по сравнению с GPU-реализацией “шагающих кубиков” без тесселяции. Созданные решения реализованы в программном комплексе, выполняющем построение и визуализацию полигональных моделей поверхностей уровня в масштабе реального времени. Разработанный комплекс был успешно апробирован на трехмерных скалярных полях насыщенности воды в образце нефтеносной породы, полученных в результате численного моделирования. Полученные изображения поверхности уровня были верифицированы с помощью системы MATLAB. В будущем планируется развитие предложенной технологии для построения и визуализации моделей рельефа местности с отрицательными уклонами.

Публикация выполнена в рамках государственного задания по проведению фундаментальных научных исследований (ГП 14) по теме (проекту) “34.9. Системы виртуального окружения: технологии, методы и алгоритмы математического моделирования и визуализации” (0065-2019-0012).

СПИСОК ЛИТЕРАТУРЫ

1. *Барладян Б.Х., Волобой А.Г., Галактионов В.А., Князь В.В., Ковернинский И.В., Солоделов Ю.А., Фролов В.А., Шапиро Л.З.* Эффективная реализация OpenGL SC для авиационных встраиваемых систем // Программирование. 2018. № 4. С. 3–10.
2. *Gavrilov N., Turlapov V.* General implementation aspects of the GPU-based volume rendering algorithm // Scientific Visualization. 2011. V. 3. № 1. P. 19–31. URL: http://sv-journal.org/2011-1/02/index.html
3. *Timokhin P., Mikhaylyuk M.* Compact GPU-based Visualization Method for High-resolution Resulting Data of Unstable Oil Displacement Simulation // Proceedings of the 29th International Conference on Computer Graphics and Vision (GraphiCon 2019), Bryansk, Russia, September 23–26. 2019. V. 2485. P. 4–6.
4. *Shakaev V.* Polygonizing volumetric terrains with sharp features // Труды 26-й Международной научной конференции GraphiCon2016, Нижний Новгород, 2016. С. 364–368.
5. *Тимохин П.Ю., Михайлюк М.В., Вожегов Е.М., Пантелей К.Д.* Технология и методы отложенного синтеза 4K-стереороликов для сложных динамических виртуальных сцен // Труды ИСП РАН. 2019. Т. 31. № 4. С. 61–72.
6. *Segal M., Akeley K.* The OpenGL Graphics System: A Specification. Version 4.6, Core Profile. The Khronos Group Inc., 2006–2018. https://www.khronos.org/registry/OpenGL/specs/gl/glspec46.core.pdf
7. *Parker S., Shirley P., Livnat Y. et al.* Interactive Ray Tracing for Isosurface Rendering // Proceedings of the IEEE Visualization 98 (VIZ’98), 1998. P. 233–238.
8. *Hadwiger M., Sigg C., Scharsach H., Bühler K., Gross M.* Real-Time Ray-Casting and Advanced Shading of Dis-

crete Isosurfaces // Computer Graphics Forum, 2005. V. 24. № 3. P. 303–312.

9. *Kim M.* GPU isosurface raycasting of FCC datasets // Graphical Models. 2013. V. 75. № 2. P. 90–101.
10. *Sanzharov V.V., Gorbonosov A.I., Frolov V.A., Voloboy A.G.* Examination of the Nvidia RTX // Proceedings of the 29th International Conference on Computer Graphics and Vision (GraphiCon 2019). 2019. V. 2485. P. 7–12.
11. *Lorensen W.E., Cline H.E.* Marching cubes: A high resolution 3D surface construction algorithm // Proceedings of the 14th Annual Conference on Computer Graphics and Interactive Techniques – SIGGRAPH'87, Anaheim, July 27–31. 1987. V. 21. № 4. P. 163–169.
12. *Bourke P.* Polygonising a scalar field, 1994. http://paulbourke.net/geometry/polygonise/.
13. *Kobbelt L., Botsch M., Schwanecke U., Seidel P.* Feature sensitive surface extraction from volume data // Proceedings of the 28th Annual Conference on Computer Graphics and Interactive Techniques, SIGGRAPH 2001. P. 57–66.
14. *Ju T., Losasso F., Schaefer S., Warren J.* Dual Contouring of Hermite Data // Proceedings of ACM Transactions on Graphics (TOG), 2002. V. 21. № 3. P. 339–346.
15. *Schmitz L., Dietrich C., Comba J.* Efficient and High Quality Contouring of Isosurfaces on Uniform Grids // Computer Graphics and Image Processing, 2009. P. 64–71.
16. *Nielson G.* Dual Marching Cubes // IEEE Visualization 2004. P. 489–496.
17. *Schaefer S., Warren J.* Dual Marching Cubes: Primal Contouring of Dual Grids // Computer Graphics Forum. 2005. V. 24. № 2. P. 195–201.
18. *Ho C.-C., Wu F.-C., Chen B.-Y., Chuang Y.-Y., Ouhyoung M.* Cubical marching squares: Adaptive Feature Preserving Surface Extraction from Volume Data // EUROGRAPHICS 2005. V. 24. № 3. P. 537–545.
19. *Manson J., Schaefer S.* Isosurfaces over simplicial partitions of multiresolution grids // Computer Graphics Forum (Proceedings of Eurographics). 2010. V. 29. № 2. P. 377–385.
20. *De Araújo B.R., Lopes D.S., Jepp P., Jorge J.A., Wyvill B.* A Survey on Implicit Surface Polygonization, 2015, ACM Computing Surveys. V. 47. № 4. P. 1–39.
21. *Matsumura M., Anjo K.* Accelerated Isosurface Polygonization for Dynamic Volume Data Using Programmable Graphics Hardware // Proceedings of SPIE-IS&T Electronic Imaging, Visualization and Data Analysis. 2003. V. 009. P. 145–152.
22. *Hansen C., Johnson C.* (editors). Visualization Handbook. Elsevier Press, 2004. 984 p.
23. *Goetz F., Junklewitz T., Domik G.* Real-Time Marching Cubes on the Vertex Shader // Proceedings of Eurographics 2005. P. 1–4.
24. *Tatarchuk N., Shopf J., DeCoro C.* Real-Time Isosurface Extraction Using the GPU Programmable Geometry Pipeline // Proceedings of ACM SIGGRAPH 2007 Courses. 2007. P. 122–137.
25. *Dyken C., Ziegler G., Theobalt C., Seidel H.-P.* High-speed Marching Cubes using HistoPyramids // Computer Graphics Forum. 2008. V. 27. № 8. P. 2028–2039.
26. *Akayev A.A, Kuzin A.K., Orlov S.G., Chetverushkin B.N., Shabrov N.N., Iakobovski M.V.* Generation of Isosurface on a Large Mesh // Proceedings of the IASTED International Conference on Automation, Control, and Information Technology (ACIT 2010), 2010. P. 236–240.
27. *Chen J., Jin X., Deng Z.* GPU-based polygonization and optimization for implicit surfaces // The Visual Computer. 2015. V. 31 (2). P. 119–130.
28. *Михайлюк М.В., Тимохин П.Ю., Мальцев А.В.* Метод тесселяции на GPU рельефа Земли для космических видеотренажеров // Программирование, 2017. № 4. С. 39–47.
29. *Bailey M., Cunningham S.* Graphics Shaders: Theory and Practice, Second Edition. CRC Press, 2011. 518 p.
30. MATLAB Documentation. Volume Visualization. https://www.mathworks.com/help/matlab/ref/isosurface.html