УДК: 004.925.83

БЫСТРОЕ УПРОЩЕНИЕ 2,5-МЕРНОЙ СЕТКИ ТРЕУГОЛЬНИКОВ С ЗАДАННОЙ ТОЧНОСТЬЮ

Алексей Юрьевич АРТЕМЬЕВ родился в 1984 г. в городе Чебоксары. Аспирант МАИ. Основные научные интересы — в области программирования трёхмерной графики (визуализация и алгоритмы), технологии разработки программ. E-mail: akydesnic@mail.ru

Alexey Yurievich Artemiev, born in 1984 in Cheboksary, Postgraduate student at the MAI. Scientific interests: 3-dimensional graphics programming (rendering and algorithms), software engineering. E-mail: akydesnic@mail.ru

Описывается метод, который позволяет быстро упростить 2,5-мерную сетку треугольников с заданной точностью. В пределах заданной точности алгоритм сохраняет геометрические особенности объекта. Особенно полезно использовать этот метод для обработки сеток, полученных после трёхмерного сканирования (например, при проверке качества деталей авиационной промышленности), моделей земной поверхности (ландшафтов) и для снижения объёма геометрических данных для передачи по сети. Описанный алгоритм можно адаптировать к многопроцессорной вычислительной среде (распараллелить).

The article describes a method for the rapid error-bounded simplification of a 2.5-dimensional triangular mesh. The algorithm presented preserves geometrical features of a surface within the given error bounds. Some method applications include post-scan mesh processing in range scanners, terrain models simplification, mesh processing in network applications. The algorithm also supports concurrent computing.

Ключевые слова: упрощение сеток треугольников, трёхмерное сканирование, моделирование земной поверхности, карта высот.

Key words: mesh simplification, range scanning, terrain modeling, height map.

1. Введение

В данной статье рассматривается метод, позволяющий быстро упростить 2,5-мерную сетку треугольников с заданной точностью. Во введении описываются основные используемые понятия и раскрываются истоки и актуальность задачи.

Трёхмерной сеткой треугольников (или просто *сеткой*) будем называть симплициальный 2-комплекс [1]. На практике сетками часто моделируют поверхности трёхмерных объектов. Важным частным случаем сеток являются *2,5-мерные сетки*. Сетка M называется 2,5-мерной, если существует плоскость P_0 с нормалью n_0, такая, что для любых двух точек m_i и m_j ($m_i \neq m_j$), принадлежащих M, существуют соответствующие им две точки p_i и p_j, принадлежащие P_0, такие что прямые, проходящие через m_i и p_i и через m_j и p_j, параллельны n_0 и $p_i \neq p_j$. Иначе говоря, M однозначно проецируется на P_0 вдоль n_0.

2,5-мерные сетки используются в инженерных программных системах. Например, съёмка трёхмерного объекта бесконтактным сканером на основе стереопары с одного положения даст 2,5-мерную сетку. Также 2,5-мерными сетками моделируют участки земной поверхности, например при съемках со спутника.

Современные сканеры обладают высокой точностью и обычно создают сетки с очень большим количеством треугольников. Например, трёхмерный сканер может создать сетку, состоящую из двух миллионов треугольников (и более). Обработка таких сеток в оперативном режиме требует больших вычислительных ресурсов, поэтому сетки необходимо упрощать, т. е. уменьшать количество треугольников, составляющих сетку.

Сетка является приближённой моделью поверхности реального объекта. При уменьшении количества треугольников создаётся ещё более грубое приближение, снижается точность представления поверхности, могут исчезать особенности её геометрии. Поэтому нужно установить допустимый предел упрощения, при котором в сетке сохраняется вся информация об исходной поверхности, необходимая для работы с её моделью.

2. Постановка задачи и допущения

Имеется 2,5-мерная сетка M_0, состоящая из k_0 треугольников.

Пусть M_k — 2,5-мерная сетка, полученная в результате упрощения M_0 и содержащая k треугольников ($k < k_0$). Тогда *ошибка упрощения* Δ_i *в точке* m_i, принадлежащей M_k, — это расстояние между m_i и ближайшей к ней точкой из M_0. Общая *ошибка упрощения* Δ определяется как максимум всех Δ_i для M_k.

Требуется максимально упростить сетку M_0 с заданной точностью, т. е. найти сетку M_k с минимальным $k < k_0$ при условии, что

$$\Delta < \varepsilon, \qquad (1)$$

где ε — *точность упрощения*, задаваемая пользователем.

Алгоритм должен работать в реальном масштабе времени. Одно из предполагаемых применений — упрощение сеток, получаемых при сканировании поверхностей пространственных объектов. Предполагается, что алгоритм будет работать одновременно со сканером в оперативном режиме. Сканер снимает объект под несколькими ракурсами, для каждого ракурса он создаёт 2,5-мерную сетку — модель соответствующих участков поверхности, после этого он собирает все фрагменты в единую трёхмерную сетку — модель объекта. Алгоритм должен упростить сетку, соответствующую одному фрагменту поверхности, за то время, которое необходимо сканеру, чтобы создать следующий фрагмент. Сканер, с помощью которого получены сетки, рассматриваемые в разд. 5, создаёт фрагмент из двух миллионов треугольников за 30 с (в среднем).

К алгоритму предъявляется ещё одно требование, обычное для алгоритмов упрощения, работающих с незамкнутыми объектами. Он должен сохранять границы сетки. Под границами понимается замыкание множества таких рёбер сетки, каждое из которых является гранью только одного треугольника той же сетки.

Предполагается, что сетка хранится в виде двух списков: списка вершин и списка треугольников. Этих данных достаточно как для описания сетки, так и для работы данного метода (в качестве исходной информации). Она же используется для представления сеток в большинстве современных графических систем (OpenGL, Direct3D, системы получения и обработки сканированных сеток) [2—4].

3. Обзор существующих методов

Сейчас существует множество различных методов упрощения сетки треугольников [5]. Кратко опишем основные из них.

Группировка вершин [6]

В методах данного подхода ограничивающую область сетки разбивают на множество областей. Вершины сетки, которые попали в одну область, сводят в единственную и удаляют вырожденные треугольники. Такой способ упрощения работает очень быстро, но не сохраняет геометрии модели (рис. 1).

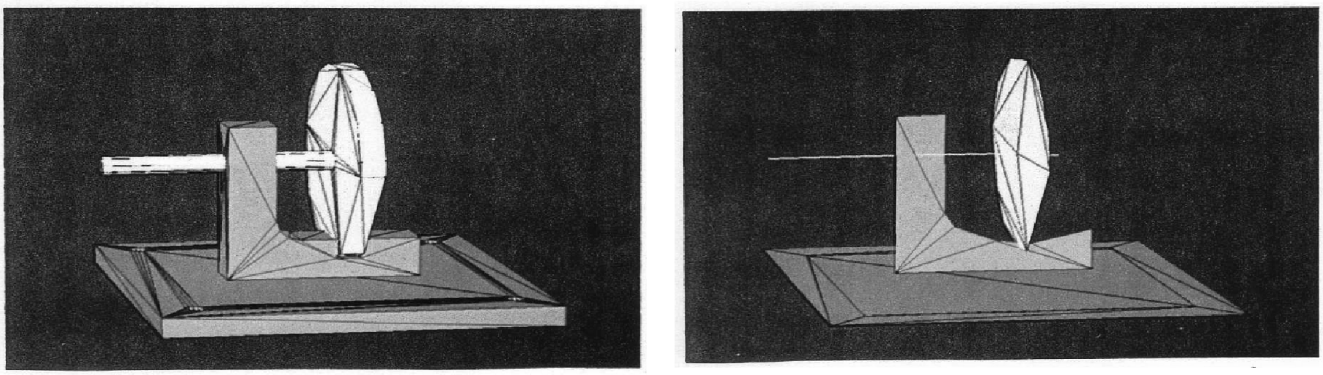

Рис. 1. Пример работы алгоритма группировки:
справа изображена исходная модель; слева — упрощённая

Явное ограничение ошибки упрощения [7]

Пользователь задаёт наибольшую ошибку упрощения (точность ε); алгоритм строит в обоих полупространствах от исходной сетки ограничивающие сетки, такие, что расстояния между точками сетки M_0 и ближайшими к ним точками ограничивающих сеток равно ε. После этого из модели последовательно удаляют элементы по некоторому критерию, следя за тем, чтобы точки получаемой поверхности находились между ограничивающими сетками (в работе [7] для этого предлагается использовать разбиение пространства на подпространства и хранение проверяемых подпространств в виде восьмеричного дерева). Данный способ строго сохраняет заданную точность, но требует больших вычислительных затрат.

Прореживание сетки [8]

Элементы сетки (вершины, рёбра или треугольники) последовательно проверяют по некоторому локальному критерию (например, накопленной квадратичной ошибке [9]) и удаляют, если они ему соответствуют. Образовавшуюся «дыру» заново триангулируют (также вместо этого иногда элемент сетки (ребро или треугольник) сводят в единственную вершину [9, 10]). Данные методы сохраняют форму модели, но обычно не могут гарантировать сохранение заданной точности, поскольку сравнивают локальную геометрию текущей итерации с моделью предыдущей, а не с исходной.

Метод, основанный на вейвлетах [11]

В данном методе сетка перестраивается так, чтобы по ней можно было построить дерево вейвлет-коэффициентов, которое позволяет легко получить упрощённую в нужной степени модель (рис. 2). Метод даёт единое представление модели, позволяющее на ходу получать варианты с разной точностью, но требует затрат по времени выполнения и памяти, не оправданных в случае, когда нужно упростить сетку один раз.

4. Описание метода

Основные шаги алгоритма упрощения (подробности и разъяснения даны в следующих разделах):

- все треугольники сетки M_0 собираются в очередь с приоритетами;
- по исходной сетке M_0 строится карта высот вдоль вектора n_0;
- последовательно рассматриваются рёбра треугольников из очереди; с помощью карты высот сравниваются локальная область вокруг ребра после его удаления и соответствующая область из M_0;
- если ребро удовлетворяет критерию, оно удаляется из сетки.

Выбор ребра для удаления

Данный алгоритм работает с представлением сетки в виде списка вершин и списка треугольников. Рёбра не хранятся явно и только при необходимости описываются через две свои вершины. Это позволяет не тратить память на хранение массива рёбер и их взаимосвязей с другими элементами сетки, а также не тратить процессорное время на обновление этих данных при изменении сетки [12].

В начале работы алгоритм заносит все треугольники в очередь с приоритетами. Приоритет треу-

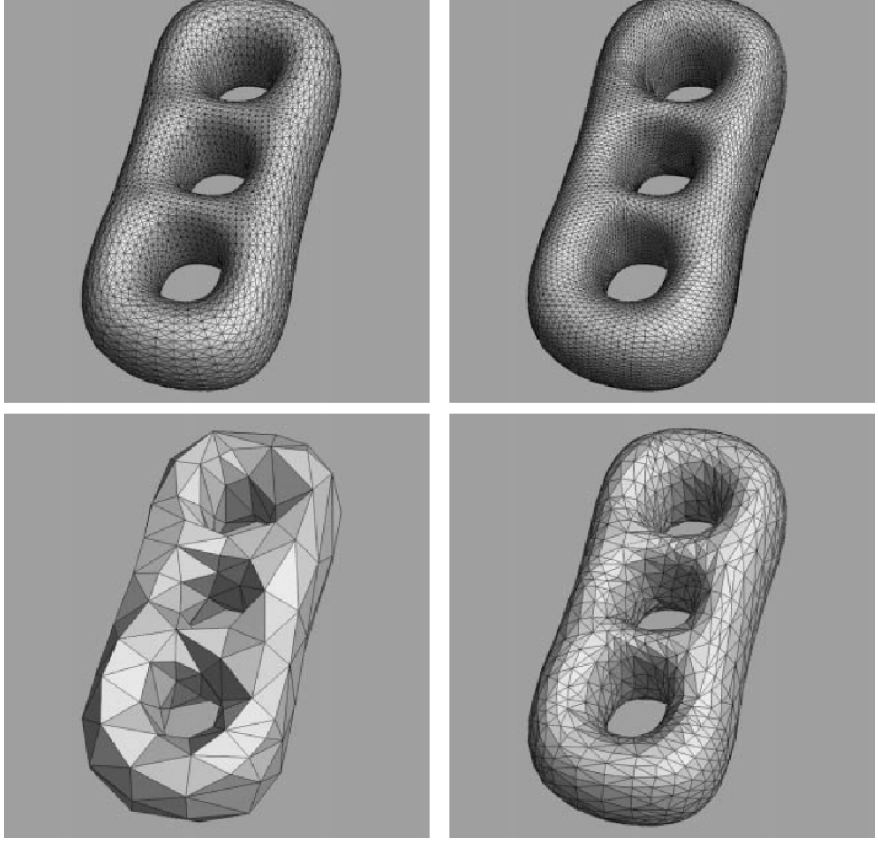

Рис. 2. Пример работы вейвлет-метода:
слева вверху показана исходная сетка; справа вверху — перестроенная;
внизу показаны два упрощённых варианта

гольника вычисляется как функция длин его рёбер. Эта функция должна быть составлена так, чтобы при обработке в первую очередь рассматривались (и удалялись) треугольники с самой малой площадью и треугольники, близкие к вырожденным (в данном случае — треугольники с самой маленькой высотой). Наличие таких треугольников делает описание поверхности неоптимальным и мешает редактировать модель как вручную, так и автоматически.

В качестве кандидатов на удаление рассматриваются рёбра треугольника, который находится в начале очереди с приоритетами.

Критерий удаления ребра

Для того чтобы сохранить границы сетки, алгоритм не рассматривает рёбра, у которых хотя бы одна вершина принадлежит границе.

Чтобы определить, можно ли удалять выбранное ребро из сетки, алгоритм использует карту высот, построенную вдоль вектора n_0.

В начале работы по всей исходной сетке строится дискретная прямоугольная карта высот (рис. 3). Шаг дискретизации карты задаётся пользователем явно или вычисляется исходя из дополнительных ограничений (например, наибольшего объёма памяти, отводимого под карту высот, или среднего количества узлов карты на треугольник исходной сетки — плотности карты высот).

Для каждого проверяемого ребра (v_i, v_j) алгоритм составляет проверяемую область из всех треугольников, у которых хотя бы одна из вершин является вершиной v_i или v_j. Затем в каждом узле карты высот, который попадает в эту область, проверяется условие

$$|h_i - h_0| < \varepsilon, \qquad (2)$$

где h_0 — значение в узле исходной карты; h_i — значение в том же узле после удаления проверяемого ребра; ε — заданная пользователем точность упрощения.

Если условие (2) выполняется во всех проверенных узлах, ребро удаляется из сетки, иначе — остаётся.

Внесение изменений в сетку

Чтобы ускорить выполнение алгоритма, ребро удаляется следующим образом [12]:

две вершины ребра сводятся в одну (в середину ребра — рис. 4);

поскольку рёбра не хранятся явно, удаление ребра сводится к удалению двух треугольников и одной вершины.

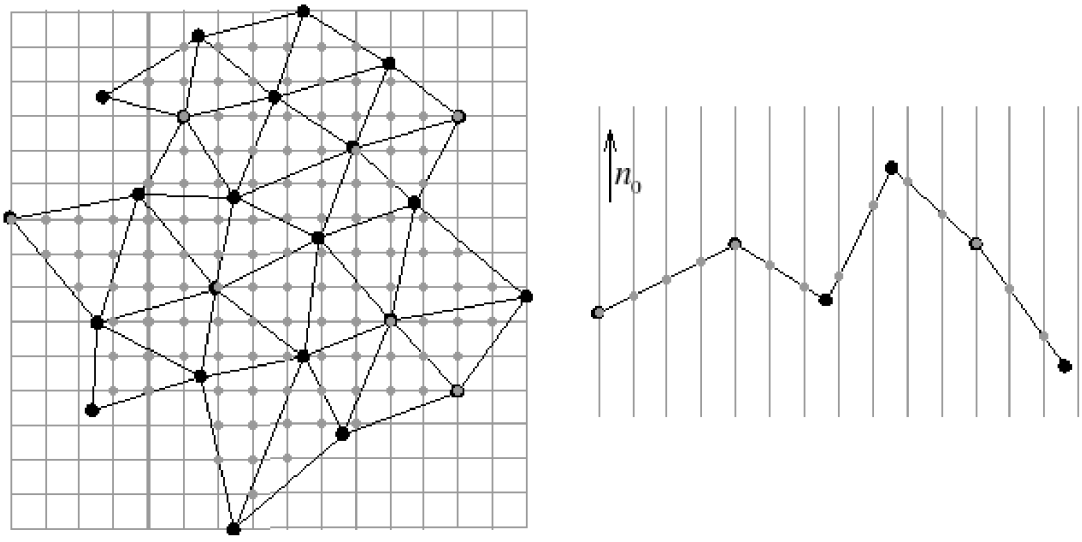

Рис. 3. Пример построения карты высот:
вектор n_0 перпендикулярен плоскости левого рисунка; чёрными линиями обозначены рёбра сетки; точками — её вершины; серым цветом нарисована карта высот; точки обозначают её узлы, которые попали внутрь сетки и могут рассматриваться при проверке рёбер

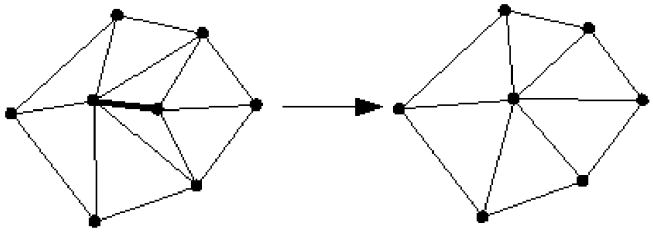

Рис. 4. Сведение ребра в вершину

5. Примеры работы алгоритма

Модели, на которых проверялся алгоритм, являются фрагментами поверхности реальных объектов, полученными с трёхмерного бесконтактного сканера. Работа алгоритма проверялась на машине со следующими параметрами: процессор — Intel Core 2 Quad, 2,4 ГГц; оперативная память — 2 Гб.

В табл. 1 указаны характеристики сеток, на которых проверялся алгоритм, параметры упрощения и основные данные статистики выполнения алгоритма.

На рис. 5—8 приведены визуальные результаты проверки алгоритма. На каждом из них слева вверху изображён фрагмент исходной модели, справа ввер-

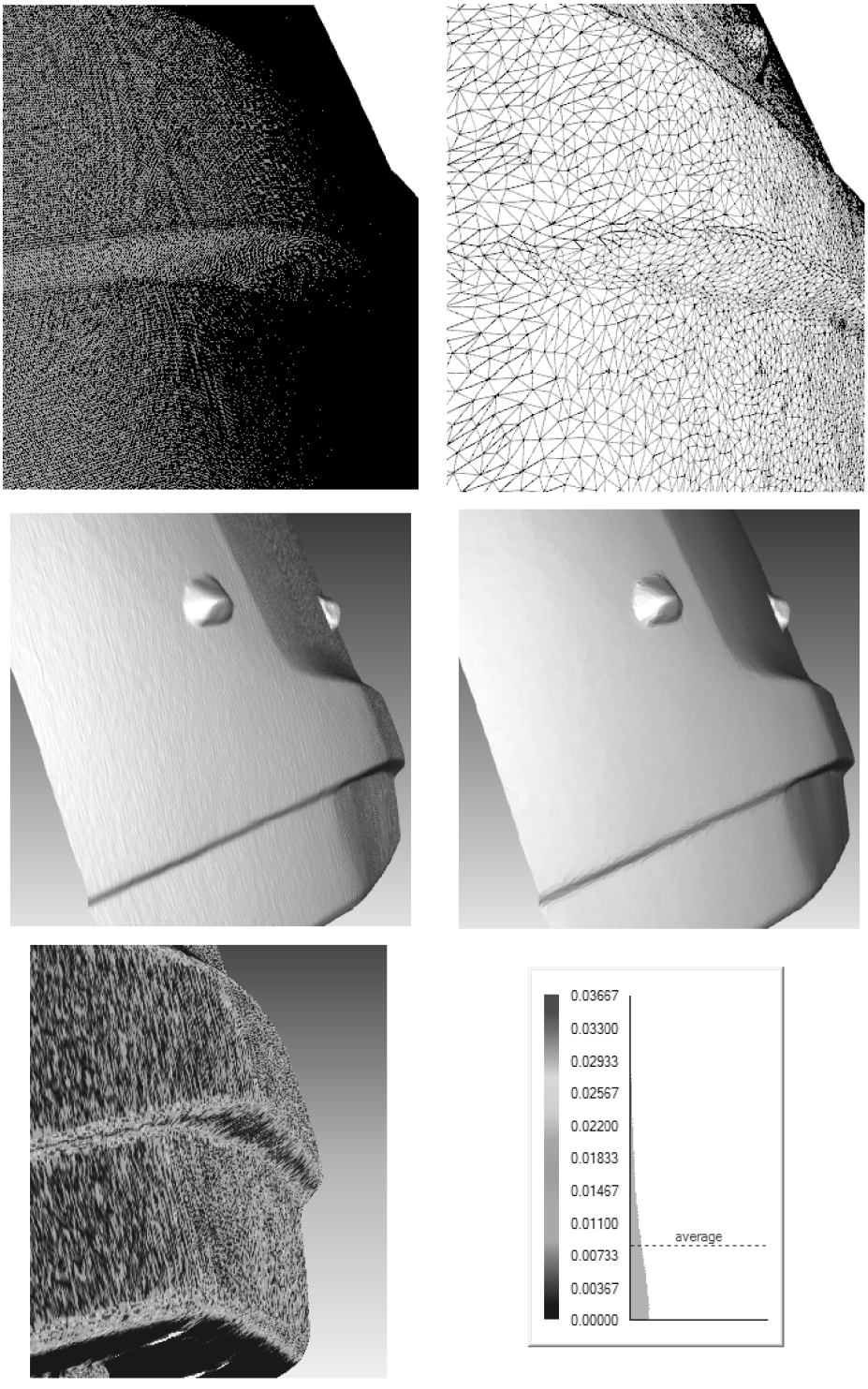

Рис. 5. Модель bumper.stl
(на средних рисунках приведены закрашенные виды:
слева — исходной, справа — упрощённой модели)

Таблица 1

Статистика работы алгоритма для разных моделей

Название файла	bumper.stl	house.stl	jaw.stl	paper.stl
Количество исход. треуг.	811911	1766192	566190	2277615
Количество упрощ. треуг.	89785	161122	66296	41503
Объём исходного файла, Кб	144335	316209	101231	409634
Объём файла результ., Кб	16038	28930	11795	7474
Заданная точность*	0,05	0,05	0,05	0,05
Наибольшая ошибка упрощения*	0,03667	0,04046	0,03856	0,02404
Средняя ошибка упрощения*	0,00843	0,00748	0,00962	0,00422
Время работы алгоритма, мс	20454	35406	9844	45562
Объём карты высот, Кб	37476	29225	11842	21370,5
Шаг дискретизации карты*	0,04	0,047	0,048	0,057

* Заданная точность, ошибки упрощения и шаг дискретизации карты даны в единицах координат вершин модели.

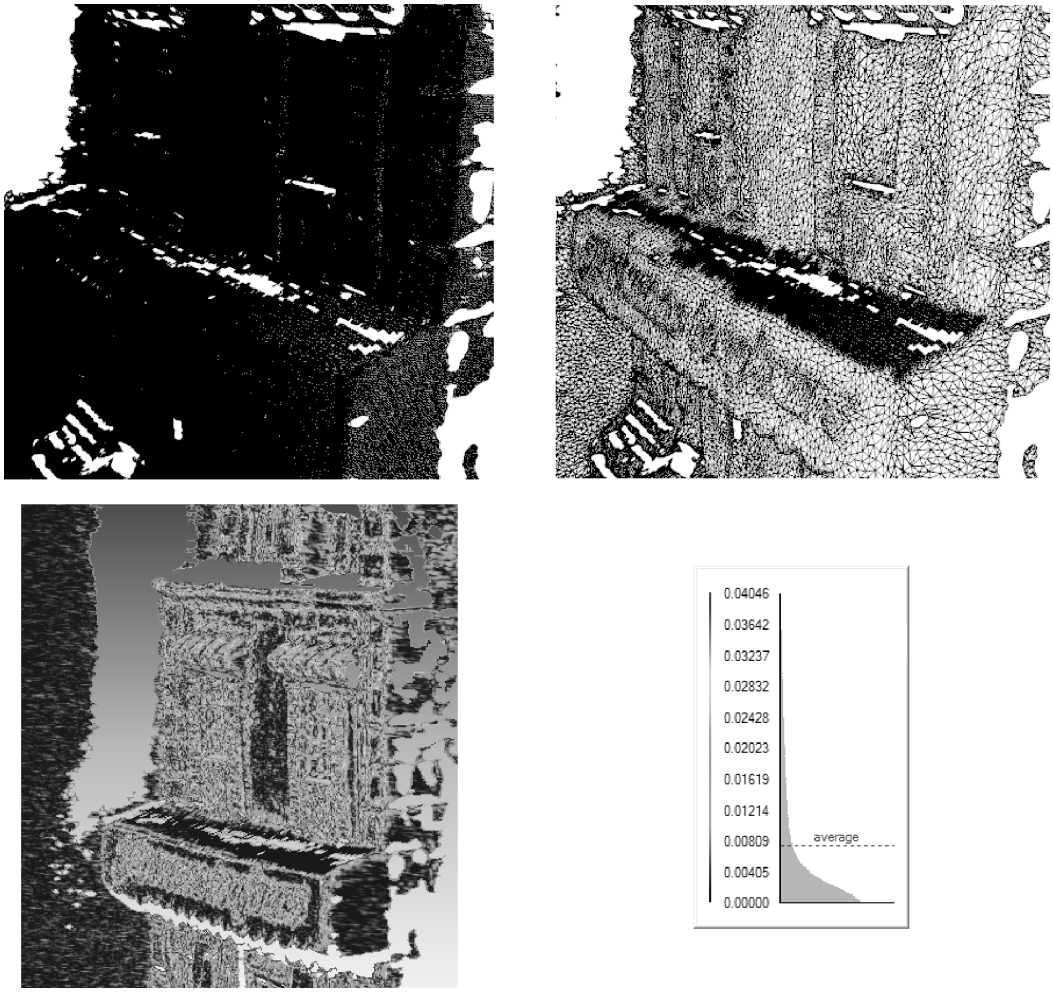

Рис. 6. Модель house.stl

ху — упрощённой, слева внизу — соответствующая диаграмма распределения ошибок, а справа внизу — распределение ошибок для всей модели.

В табл. 2 собраны сведения по упрощению одной из моделей при разных значениях управляющих параметров (в таблице они выделены полужирным шрифтом).

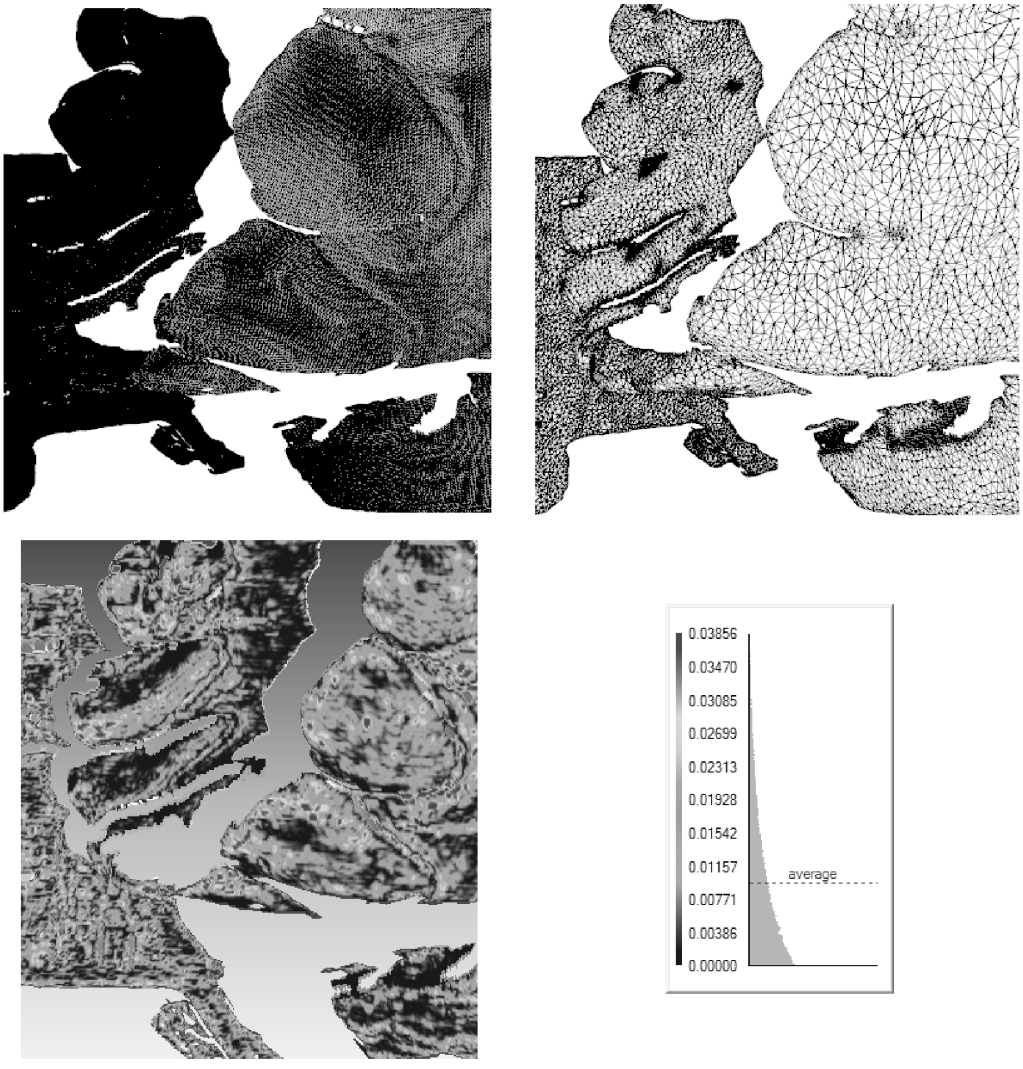

Рис. 7. Модель jaw.stl

Таблица 2

Сравнение работы алгоритма при разных параметрах упрощения (модель house.stl: 1766192 треугольников, 316209 Кб)

Заданная точность*	0,05	0,05	0,05	0,01	0,1
Наибольшая ошибка упрощения*	0,04046	0,03976	0,03976	0,00919	0,07325
Средняя ошибка упрощения*	0,00748	0,00728	0,00708	0,00243	0,00998
Шаг дискретизации карты*	0,047	0,03	0,021	0,021	0,021
Объём карты высот, Кб	29225	73053	146102	146102	146102
Время работы алгоритма, мс	35031	46969	66015	62203	66969
Количество упрощ. треуг.	161122	169746	184212	385624	171374
Объём файла результата, Кб	28930	30474	33063	69274	30746

* Заданная точность, ошибки упрощения и шаг дискретизации карты даны в единицах координат вершин модели.

6. Выводы

Достоинства метода

Описанный способ позволяет быстро упростить 2,5-мерную сетку (например, фрагмент поверхности, полученный с бесконтактного трёхмерного сканера или модель земной поверхности), что допускает его использование в критичных ко времени приложениях, например при сканировании больших объектов, таких, как самолет. Быстрота алгоритма — следствие нескольких факторов:

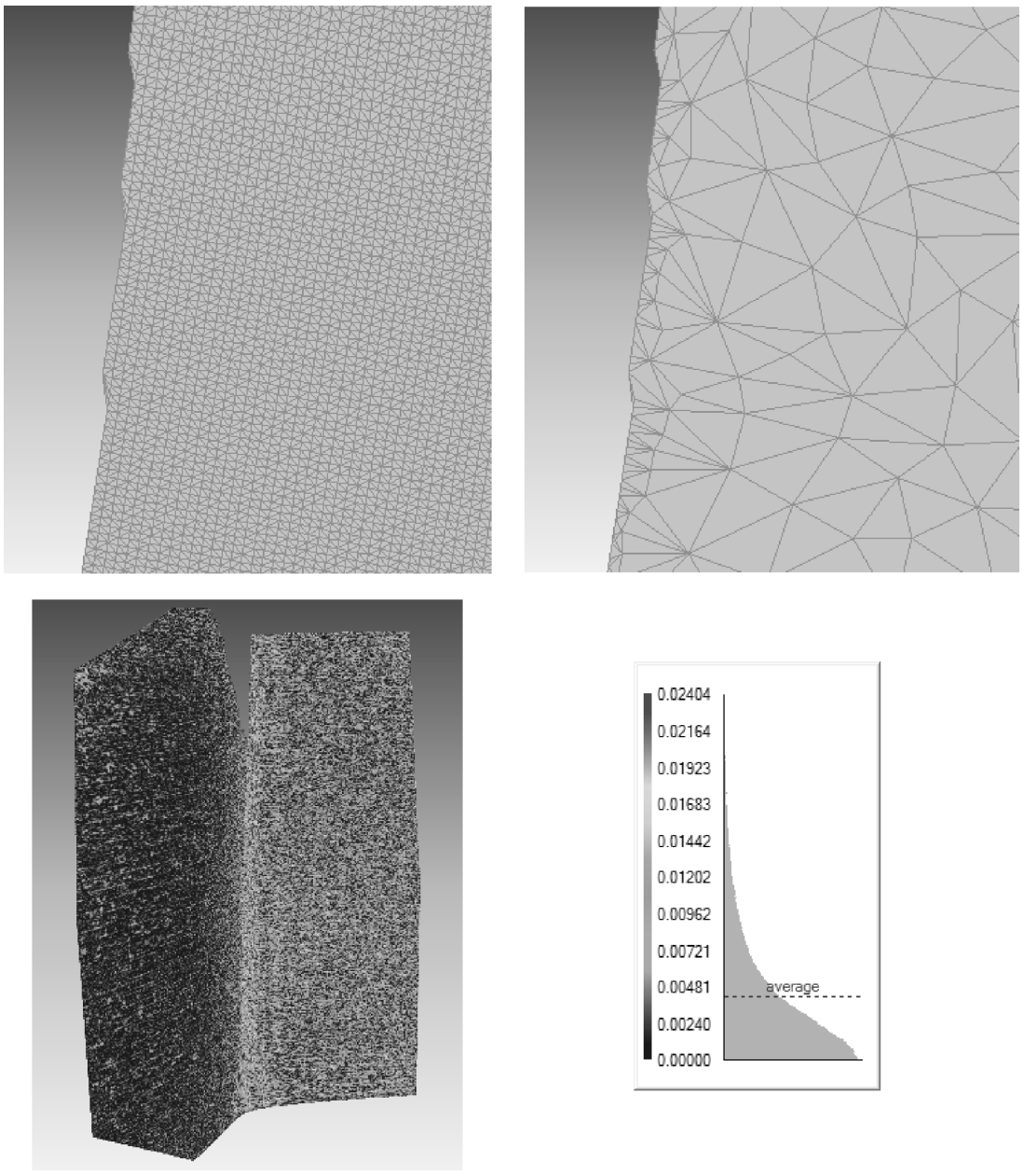

Рис. 8. Модель paper.stl:
на нижнем левом изображении показана диаграмма для всей модели

• быстрый выбор удаляемого ребра: используемая структура данных (очередь с приоритетами) позволяет улучшить результат и является очень эффективной: операции над ней выполняются за время O(log n);

• быстрая проверка элементов сетки: критерий, описанный в разд. 4, прост и выполняется очень быстро;

• быстрое обновление сетки: обновление сетки на одной итерации алгоритма сводится к удалению двух треугольников и одной вершины (плюс обновление связей между оставшимися вершинами и треугольниками). При эффективной реализации соответствующих структур данных эта операция также оказывается максимально эффективной. Лишние вспомогательные данные не хранятся и, соответственно, не обрабатываются.

Если истинное расстояние между сетками (в указанной точке) равно d_0, то алгоритм измерит расстояние d, которое всегда больше истинного либо равно ему; если d меньше ε, то и d_0 меньше ε, поэтому алгоритм в большинстве случаев сохраняет точность «с запасом».

Недостатки метода

В предложенном методе сохранение точности зависит от плотности карты высот. Чем меньше шаг дискретизации карты, тем точнее будет результат;

если шаг слишком большой, некоторые треугольники могут оказаться между узлами сетки, что может привести к потере точности.

В данном методе как выигрыш в скорости работы, так и достижение необходимой точности упрощения достигаются во многом за счёт затрат по памяти: чем плотнее карта высот, тем больший объём памяти она занимает.

Возможные пути развития алгоритма

Алгоритм работает с сеткой локально, поэтому выделять память сразу для всей карты высот необязательно. Можно создавать отдельные участки карты при необходимости и удалять, если они больше не нужны. Но при таком подходе необходимо найти компромисс между снижением расхода памяти и снижением скорости из-за создания и удаления (или замещения) участков карты в процессе работы. Возможным вариантом реализации является буфер (кэш, пул) участков карты высот. Потери в скорости работы можно компенсировать параллельной обработкой независимых участков сетки. При параллельной обработке придётся также строить отдельные очереди с приоритетами (и, возможно, некоторые другие структуры данных) для каждой локальной области, поскольку совместное использование вычислительными потоками одного блока данных приведёт скорее не к увеличению, а к уменьшению скорости работы из-за синхронизации обращений к данным.

Ещё один вариант ускорения работы алгоритма — подобрать оптимальный порядок проверки узлов сетки внутри области. Это может дать выигрыш на областях, которые не проходят критерий карты высот (алгоритм может быстрее найти точку, не соответствующую критерию, и, соответственно, быстрее отбросить данную область).

Библиографический список

1. Комплекс (матем.) // Большая советская энциклопедия. — 3-е изд. — М., 1978.

2. *Ву М., Нейдер Дж., Девис Т., Шрайнер Д.* OpenGL. Официальное руководство программиста. — СПб.: ООО «ДиаСофтЮП», 2002.

3. *Порев В.Н.* Компьютерная графика. — СПб.: БХВ-Петербург, 2004.

4. *Шикин А.В., Боресков А.В.* Компьютерная графика. Полигональные модели. — М.: Диалог-МИФИ, 2005.

5. *Cignoni P., Montani C., Scopigno R.* A Comparison of Mesh Simplification Algorithms // Computers & Graphics. — 1998. — V. 22. — No. 1 (Feb.). — P. 37-54.

6. *Rossignac J., Borrel P.* Multi-resolution 3D approximations for rendering complex scenes // Modeling in Computer Graphics: Methods and Applications / Ed. by Falcidieno B., Kunii T. — 1993. — P. 455-465.

7. *Cohen J., Varshney A., Manocha D., Turk G., Weber H., Agarwal P., Brooks F., Wright W.* Simplification envelopes // Computer Graphics (SIGGRAPH'96 Proc.). 1996. — P. 119-128.

8. *Schroeder W.J., Zarge J.A., Lorensen W.E.* Decimation of triangle meshes // Computer Graphics (SIGGRAPH'92 Proc.). — 1992. — No. 26(2). — P. 65-70.

9. *Garland M., Heckbert P.S.* Surface Simpli?cation Using Quadric Error Metrics // Computer Graphics (SIGGRAPH'97 Proc.). — 1997. — P. 209-218.

10. *Hoppe H.* Progressive meshes // Computer Graphics (SIGGRAPH'96 Proc.). — 1996. — P. 99-108.

11. *Eck M., DeRose T., Duchamp T., Hoppe H., Lounsbery M., Stuetzle W.* Multiresolution Analysis of Arbitrary Meshes // Computer Graphics (SIGGRAPH'95 Proc.). — 1995. — P. 173-182.

12. *Артемьев А.Ю.* Быстрое удаление элементов сетки треугольников для алгоритма прореживания (децимации) сетки // Материалы XVI Международной конференции по вычислительной механике и современным прикладным программным системам (ВМСППС'2009), 25-31 мая 2009 г., Алушта. — М.: Изд-во МАИ-ПРИНТ, 2009. — С. 62-64.

Московский авиационный институт
Статья поступила в редакцию 16.11.2009