Метод сжатия полигональных сеток с потерями*

А.А. Зачесов
azachesov@artec-group.com
Институт Прикладной Математики им. М.В. Келдыша, Москва, Россия

С развитием и повсеместным распространением технологий трехмерного сканирования происходит экспоненциальный рост объема получаемых в результате сканирования данных – главным образом, трехмерных сеток, хранящих геометрические данные снимавшегося объекта. Становится актуальной проблема эффективного хранения этих данных с минимальными потерями или без потерь их точности. Предлагается алгоритм сжатия с потерями для трехмерных полигональных сеток, полученных методом трехмерного сканирования. Описываются характеристики этого алгоритма. Приводится сравнение объема данных, получаемых в результате применения предложенного алгоритма, с популярными форматами для хранения трехмерных сеток.

Ключевые слова: Сжатие с потерями, трехмерная модель, трехмерная сетка, квантование

Lossy compression method for polygonal meshes*

A.A.Zachesov
Keldysh Institute of Applied Mathematics, Moscow, Russia

Development and spreading of three-dimensional scanning technologies result in exponential growth of scanned data volumes – mainly three-dimensional grids that store geometry information. It becomes an actual problem to efficiently store this data with minimal or without precision loss. We propose a lossy compression algorithm for three-dimensional polygon meshes obtained by three-dimensional scanning. We describe algorithms' characteristics and show the comparison of the compression ratio for the proposed algorithm and popular file formats for storing three-dimensional meshes.

Keywords: Lossy compression, three-dimensional mesh, quantization

Все способы получения трехмерных моделей можно разделить на две группы: моделирование, когда создание геометрии модели происходит в полуавтоматическом режиме, и реконструкция, когда построение геометрии осуществляется автоматически. Это может быть реконструкция трехмерной модели из многоракурсного видео, реконструкция по структурированной подсветке или реконструкция из видеоряда с одной камеры. Любой из этих методов осуществляет построение трехмерной сетки с некоторой погрешностью относительно реальных размеров исходных объектов.

Данные, получаемые в результате трехмерного сканирования, как правило, являются промежуточными перед дальнейшим этапом обработки. Это означает, что точность промежуточных данных влияет на качество этой дальнейшей обработки, из-за чего для хранения данных, как правило, используются форматы, предполагающие хранение данных без потерь. Это приводит к значительному объему хранимых промежуточных данных. При этом, как говорилось выше, данные изначально содержат погрешность, которой можно пренебречь. Это позволяет создать алгоритм сжатия данных с потерями, позволяющий существенно уменьшить объем хранимых данных.

Работа опубликована при финансовой поддержке РФФИ, грант 15-07-20347

Обзор существующих методов

Существует значительное количество методов, осуществляющих сжатие произвольных полигональных сеток с потерями [2, 3]. Их можно разделить на две большие группы: методы, использующие упрощение исходной сетки – уменьшение количества полигонов – и методы, сохраняющие исходную структуру сетки (количество вершин и треугольников без изменений). Методы из первой группы позволяют достичь низких коэффициентов сжатия, отбрасывая частью информации об исходных полигонах. В [2] уменьшается количество полигонов трехмерной сетки путем объединения нескольких полигонов в один. Также описывается подход, основанный на перестраивании геометрической структуры модели с целью получения равномерной сетки.

При втором подходе структура (количество вершин и треугольников) трехмерной сетки остается неизменной. В [5] используется квантование координат, а также замена значений координат их разностью. Для квантования координат используется фиксированное число бит. В [1] для кодирования значений разности координат используются коды Хаффмана. Для представления информации о треугольниках исходной трехмерной сетки чаще всего используется информация об их связности [4]. При использовании такого подхода для сохранения информации об одном треугольнике достаточно 2 бит.

Предложенный алгоритм

Предлагается алгоритм сжатия с потерями для трехмерной полигональной сетки, описываемой совокупностью массива вершин и объединяющих их треугольников. Каждая вершина описывается тремя вещественными координатами, а каждый треугольник – тремя целочисленными индексами вершин. Выходными данными алгоритма является сжатое представление исходной сетки, которое далее также будет именоваться выходным потоком. Исходные данные можно также представить как последовательность вершин и последовательность треугольников, которые объединяют эти вершины. Рассмотрим обработку этих наборов данных отдельно.

Обработка последовательности координат вершин

Рассмотрим одну из вершин последовательности, например, первую. Координаты этой вершины по каждой из осей могут значительно отличаться друг от друга по абсолютным значениям. В то же время значения координат по каждой из осей будут близки к значениям координат соседней вершины по той же оси. Будем обрабатывать последовательности координат по каждой из осей независимо, а для определенности в дальнейшем рассматривать сжатие последовательности координат по оси x. Для последовательностей координат по остальным осям обработка аналогична.

Основной идеей алгоритма сжатия последовательности координат, как и в методе, описанном в [5], является замена значений координат значениями разницы между соседними координатами. Сжатие достигается за счет квантования значений разницы и использования для них меньшего количества бит, чем для значения исходной координаты. Для каждой вершины в выходной поток записывается значение $\Delta = x_n - x_{n-1}$ с заданной точностью. В отличие от сжатия изображений или видео, диапазон значений Δ не ограничен, поскольку абсолютные значения исходной сетки не ограничены, кроме того, сама исходная сетка может содержать в себе разрывы. Для ограничения диапазона изменения значений разницы координат вводится параметр максимальной кодируемой разницы, или порога квантования, задаваемый однократно для всей последовательности координат по одной оси. В рамках этого диапазона для значений разницы используется операция квантования по уровню с заданным количеством бит. Для каждой координаты ее значение в выходном потоке аппроксимируется, как:

$$x_n \cong x'_n = x'_{n-1} + \Delta. \qquad (1)$$

где x'_{n-1} – значение координаты для предыдущей вершины после применения к ней аналогичной аппроксимации. Для первой координаты $x_0 = x'_0$, и ее значение помещается в выходной поток «как есть». Значение Δ есть аппроксимация исходной разницы между $x_(n-1)'$ и x_n, вычисляемое как:

$$\Delta = \left(\frac{x'_n - x'_{n-1}}{Q} \cdot 2^p \right). \qquad (2)$$

где Q – порог квантования, p – количество уровней квантования, которые являются параметрами метода.

Использование координат после их аппроксимации в качестве базовых для кодирования следующих координат в последовательности необходимо для корректной распаковки. Если разница текущей рассматриваемой координаты с предыдущей координатой больше заданного порога квантования, текущая координата помещается в выходной поток «как есть». Такие вершины назовем ключевыми. Всю последовательность исходных координат можно разбить на набор независимых последовательностей, состоящих из ключевой вершины и произвольного количества координат, закодированных разницей.

Оценка коэффициентов сжатия для последовательности координат

Для оценки коэффициентов сжатия будем считать, что для сохранения вещественного значения координаты необходимо 4 байта. Коэффициенты сжатия будем рассматривать только для последовательности координат по одной из осей. Лучший коэффициент сжатия достигается для последовательности координат, в которой все расстояния между соседними вершинами меньше, чем выбранный порог квантования (то есть ключевая координата одна), а для квантования используется два уровня, то есть для сохранения значения разницы достаточно одного бита. Пусть размер исходных данных составляет $32 \cdot n$ бит, где n – количество координат в последовательности. Тогда объем сжатых данных, пренебрегая объемом, необходимым для сохранения ключевой вершины, составит $2 \cdot n$ бит, коэффициент сжатия – 6,25%

Худший коэффициент сжатия достигается для последовательности координат, все расстояния между соседними вершинами которой больше, чем выбранный порог квантования (то есть все координаты являются ключевыми). В этом случае объем увеличится на один бит для каждой ключевой вершины, то есть станет равным $33 \cdot n$ бит, коэффициент сжатия – 103,125%.

Алгоритм выбора оптимальных параметров сжатия

Степень сжатия зависит от выбора параметров сжатия: порога квантования и количества уровней квантования, причем максимальная степень

сжатия достигается, если порог квантования выбрать неограниченно большим, а количество уровней квантования ограничить двумя. Однако это приведет к значительным потерям точности исходных данных, что сводит на нет полезность сжатия в целом. Выбор некоторых универсальных параметров невозможен, поскольку характер исходных данных может сильно варьироваться. Для контроля степени сжатия предлагается метод автоматического подбора оптимальных параметров (при которых размер сжатого представления исходных данных минимизируется) с ограничением погрешности значений вершин после сжатия относительно исходных значений.

Размер данных после сжатия всегда определяется количеством ключевых вершин и количеством уровней квантования, используемых для кодирования значения разницы вершин. Оптимальными параметрами сжатия являются такие Q и p, при которых размер файла S минимален. Функция для минимизации выглядит следующим образом:

$$S = keys \cdot 33 + (n - keys) \cdot (p + 1). \qquad (3)$$

Здесь $keys$ – количество ключевых координат, n – общее количество координат, p – используемое количество уровней квантования. Прибавление единицы в $p + 1$ позволяет учесть влияние бита-флага на объем файла. По той же причине используется множитель $33 = 32 + 1$. Количество ключевых вершин $keys$ может быть вычислено для любого фиксированного значения порога квантования Q. При добавлении ограничения на допустимую ошибку относительно исходных значений появляется связь между порогом квантования и количеством необходимых уровней квантования

$$p = \lceil \log_2(\frac{Q}{e}) \rceil. \qquad (4)$$

где e – значение допустимой погрешности. Действительно, чтобы удовлетворить требование о допустимой погрешности необходимо и достаточно, чтобы интервал, получаемый после квантования имел длину меньше e. Отсюда следует, что при увеличении Q растет p, и наоборот, а также что для минимизации функции достаточно осуществить перебор по Q.

Обработка последовательности индексов треугольников

Кроме координат вершин для восстановления трехмерной сетки необходима также информация о соединяющих их треугольниках. Как правило, для каждого треугольника сохраняются три индекса вершин, которые составляют этот треугольник, то есть исходными данными является последовательность троек целочисленных индексов. Эта последовательность построена таким образом, чтобы соседние тройки в последовательности соответствовали соседним в пространстве треугольникам и имели общие индексы вершин. Такая очередность позволяет построить алгоритм генерации последовательности на основе рекуррентных формул, использующим зависимости между значениями индексов.

В силу наличия разрывов в исходной трехмерной сетке невозможно закодировать всю последовательность треугольников, используя единственную рекуррентную формулу. Вид зависимостей индексов также может отличаться. Для борьбы с этим алгоритм использует несколько предзаданных рекуррентных формул вместо одной, а также вводится понятие «ключевой тройки индексов» – тройки, которая не может быть предсказана из предыдущей при использовании любой рекуррентной формулы. Для каждой ключевой тройки существует подпоследовательность троек, выводимых из нее применением одной и той же рекуррентной формулы (длина этой последовательности может быть нулевой). Каждая подпоследовательность сохраняется в выходной поток следующим образом: ключевая тройка индексов, количество выводимых из нее других троек и код используемой для восстановления рекуррентной формулы.

Пример рекуррентной формулы, отражающей зависимости между индексами вершин соседних треугольников

$$i_n = i_{n-1}; j_n = j_{n-1} - 1; k_n = j_{n-1}; n = 2m$$

$$i_n = i_{n-1} + 1; j_n = j_{n-1} + 2; k_n = i_{n-1} + 2; n = 2m + 1$$

где i, j, k – индексы вершин треугольников. Данная рекуррентная формула выведена эмпирическим путем при анализе исходных трехмерных сеток. Всего используется 4 рекуррентные формулы аналогичного вида.

Оценка коэффициентов сжатия для последовательности индексов треугольников

Рассмотрим лучший и худший коэффициенты сжатия для последовательности троек индексов. Будем считать, что каждая тройка индексов требует 12 байт (три целочисленных значения). Длину последовательности будем кодировать двумя байтами. Для поддержки возможного увеличения количества используемых рекуррентных формул, будем считать, что для сохранения типа формулы необходим 1 байт. Лучший коэффициент достигается, если все треугольники исходной последовательности предсказываются с использованием одной из рекуррентных формул. В этом случае одна подпоследовательность из 2^{16} троек индексов – 768 кбайт

Таблица 1: Сравнение объема данных

Используемый формат данных	Объем данных относительно исходного
Object file формат (текстовый)	240%
Ply формат (бинарный)	102%
Ply формат + LZMA	48,8%
Метод, описанный в [1]	17%
Предложенный алгоритм (допустимая погрешность 0.1 мм)	13,7%
Предложенный алгоритм (допустимая погрешность 2 мм)	8,9%
Предложенный алгоритм + LZMA (допустимая погрешность 2 мм)	3,4%

– может быть представлена как 15 байт (исходная ключевая тройка индексов, длина последовательности и тип рекуррентной формулы). Коэффициент сжатия составляет 0,002%. Худший коэффициент достигается в случае, если исходная трехмерная сетка представляет собой набор несвязанных друг с другом треугольников. В этом случае каждая тройка попадает в выходной поток как есть, и коэффициент сжатия равен 125% (каждая тройка занимает 15 байт вместо исходных 12). Среднее количество предсказанных индексов треугольников для нескольких сотен исходных сеток, использованных для тестирования алгоритма, составляет 98% от общего числа, то есть средний коэффициент сжатия последовательности индексов треугольников составляет 2,5%.

Дальнейшее увеличение степени сжатия

Размер блока данных, получаемый в результате сжатия предложенным алгоритмом, может быть дополнительно уменьшен применением к нему алгоритма сжатия без потерь, например, алгоритма LZMA. Действительно, при сжатии координат в случае, если значения соседних координат отличаются друг от друга менее чем на $\frac{e}{2}$, где e – допустимая погрешность координат, индекс квантования, используемый для их сохранения, будет одинаков. Наличие в выходном потоке подобных повторов позволяет уменьшить объем файла.

Выводы

В таблице 1 приведено сравнение объема данных, получаемых при сохранении трехмерной сетки в широко распространенные форматы с предложенным алгоритмом. За 100% принят объем данных, рассчитанный тем же способом, который использовался для оценки лучшего и худшего коэффициентов сжатия. Объем, занимаемый координатами вершин и индексами треугольников, суммируется. Значения в таблице являются усредненными по нескольким сотням трехмерных сеток, использованных для тестирования алгоритма. В таблице приведены результаты для сжатия с несколькими различными значениями погрешности (значения координат заданы в миллиметрах).

Представленный алгоритм используется для хранения и передачи по сети данных, полученных трехмерным сканированием. Используется сжатие с допустимой потерей точности 0,1 мм при точности исходных отсканированных данных 2 мм. В соответствии с таблицей это позволяет уменьшить объем хранимых данных в 7-8 раз. Новизна и отличие предложенного метода относительно существующих методов состоит в использовании рекуррентной формулы для восстановления последовательности треугольников, а также использование автоматического выбора параметров при сжатии последовательности вершин.

Описанный алгоритм сжатия трехмерных сеток не поддерживает обработку сеток, которые имеют привязанную к ним текстуру, а именно сохранение информации о текстурных координатах. Дальнейшее развитие алгоритма состоит в добавлении поддержки текстурных координат. Они могут быть сжаты аналогично геометрическим координатам. В качестве допустимой погрешности может использоваться точность значения координаты, измеряемая в пикселях исходного файла текстуры.

Литература

[1] *Deering M.* Geometry compression // Proceedings of SIGGRAPH Conference, 1995. – pp.13-20.

[2] *Gotsman C., Gumhold S., Kobbelt L.* Simplification and Compression of 3D Meshes // Springer. Tutorials on Multiresolution in Geometric Modelling. – 2002. – C.319-361.

[3] *Rossignac J.* 3D mesh compression // Visualization Handbook. – 2003. – C.359-379.

[4] *Rossignac J.* Edgebreaker: Connectivity Compression for Triangle Meshes // IEEE Trans. Visualization and Computer Graphics, 1999. – vol. 5, no. 1 – pp.47-61.

[5] *Touma C., Gotsman C.* Triangle Mesh Compression // Proceedings of Graphics Interface, 1998.