Специализированная Система Визуализации Некоторых Задач Оптимального Управления

А.В. Мошков, В.Ю. Пахотинских, В.О. Решетняк
ИММ УрО РАН

moshkov@list.ru, reshetnyakv@yandex.ru.

Abstract

В работе описывается специализированная система визуализации для одной из задач оптимального управления. Основное внимание уделяется методам обработки и хранения больших объемов данных с применением фильтрации по средством конвертирования в различное представление (воксельное и полигональное) и его последующей визуализации. При этом сохраняется внутренняя и внешняя структура объектов. Отдельно рассматривается возможность задания степени детализации для обеспечения гибкости решения задачи.
Keywords: оптимальное управление, трехмерная визуализация, воксельная графика, полигональная графика, триангуляция, октарное дерево.

1. ВВЕДЕНИЕ

Поиск областей достижимости в задачах оптимального управления послужил источником нашей постановки. Математику необходимо изучать общий вид и внутреннюю структуру области достижимости, причем огромный объем полученного расчетного материала требует использования средств трехмерной визуализации.

Ряд причин, связанных с реализацией алгоритма, привел к тому, что вычисленные данные хранятся в формате *.bmp, при этом входные данные составляют порядка миллионов, а то и миллиардов точек. Что и породило комплекс проблем, решенных нами при разработке системы.

В результате создан комплекс программ, позволяющий провести фильтрацию данных не искажающую ни внешнюю и ни внутреннюю структуру объекта. И позволяет отобразить как процесс построения, так и сам объект в разных ракурсах.

2. ОБЩАЯ СТРУКТУРА ПРОГРАММНОГО КОМПЛЕКСА

Данный программный комплекс состоит из набора утилит для работы с огромным облаком точек и его последующей визуализации. В разработке использованы технологии OpenGL и DirectX.

Стадии конвейера:

- Обработчик bitmap файлов
- Вычислитель освещенности
- Конвертер в воксельный формат и создание структур хранения сцены
- Сглаживатель воксельных объектов
- Конвертер в полигональный формат (триангулятор)

3. ВХОДНЫЕ ДАННЫЕ

На вход поступает набор точек в трехмерном пространстве, заданный в виде плоских срезов по оси Z, и хранимые в файлах формата *.bmp.

Преимущества хранения в *.bmp: если информацию о точке хранить в виде координат (x,y,z) типа double, то информация об одной точке будет занимать 24 байта. Мы использую монохромные bitmap, поэтому объем информации о точке можно считать приблизительно равным 1 биту (если отбросить заголовок *.bmp файла и пустое пространство в файле). Таким образом с использованием bitmap файлов объем хранимой информации уменьшается в 24*8=192 раза. На самом деле это достаточно грубая оценка, так как заполняемость bitmap файла точками зависит от конкретной системы.

Если использовать сжатие файлов (например, встроенное сжатие файлов NTFS), то потребуется еще меньше дискового пространства для сохранения информации о состоянии системы. Следует отметить, что степень сжатия монохромных bitmap файлов очень высокая, в отличие от бинарных файлов.

4. ПРЕДОБРАБОТКА ДАННЫХ

4.1. Сортировка и нормирование

Полученный файл подвергаются обработке. Последовательно считываются все bitmap файлы и «на лету» сортируются в нужной последовательности и нормируются. Получаем файл с данными, подготовленными для дальнейшей обработки, содержащий тройки координат (x,y,z).

4.2. Вычисление освещенности

Для улучшения восприятия трехмерности представления сцены используется освещение.

Освещенность рассчитывается методом интерполяции векторов нормалей. Вектор нормали в точке является вектором нормали аппроксимирующей плоскости, соседних точек, лежащих на определенном расстоянии от заданной точки. Расстояние является переменная величина. В зависимости от изменения величины этой переменой меняется степень освещенности.

Полученный файл поблочно считывается в оперативную память с использованием технологии FileMapping, после чего в этом блоке рассчитывается освещенность, и весь блок вместе с полученными значениями освещенности записывается в бинарный файл. В конце данной операции получаем сжатый файл, в котором хранятся четверки чисел (x, y, z координаты и цвет).

5. ПРЕОБРАЗОВАНИЕ В ВОКСЕЛЬНЫЙ ФОРМАТ

Полученный бинарный файл данных считывается и данные помещаются в воксельный куб с заданным размером.

Для хранения исходной сцены была выбрана структура хранения данных, называемая Octree (октарное дерево). Октарное дерево подходит именно для реализации этой задачи, так как оно имеет преимущество для открытых пространств, какое имеется в нашем случае. Октарное дерево представляет изображение на уровне объектов или вокселей. В нашем случае используется воксельная графика. От размера куба зависит качество полученного объекта. На данном этапе уже доступна возможность визуализации. Размер куба влияет на качество графического представления, чем он выше, тем качественнее, но увеличиваются затраты ресурсов. Компромиссное решение находится от требования задачи и имеющегося оборудования.

6. ПЕРЕВОД В ПОЛИГОНАЛЬНЫЙ ВИД И СГЛАЖИВАНИЕ

На следующем шаге происходит сглаживание воксельного объекта для последующей триангуляции. Триангуляция проходит несколькими этапами. На каждом этапе происходит уменьшение количества полигонов (треугольников), причем число этапов зависит от степени детализации. Чем выше уровень детализации, тем больше размер получаемого файла.

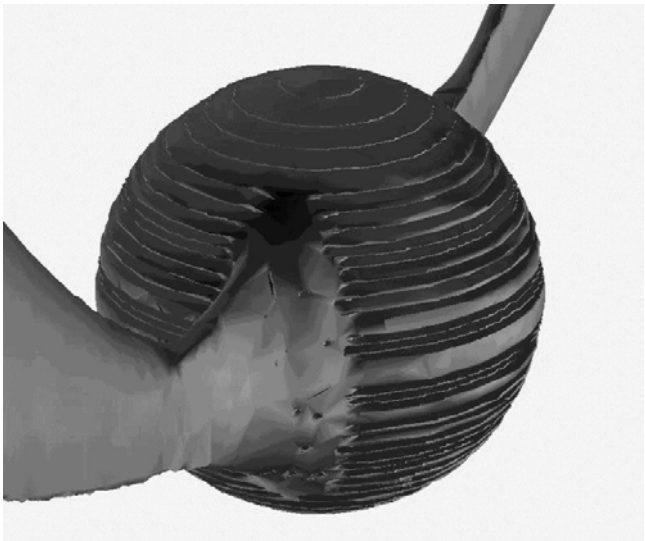

Рисунок 1: Средний уровень детализации полигонального объекта.

В итоге получаем разреженный набор треугольников, который позволяет увидеть на экране более качественную картинку, чем в воксельном представлении. К тому же размер файла данных значительно меньше. После всех операций имеется возможность сохранить данные в формате *.obj для просмотра в любых других трехмерных графических пакетах.

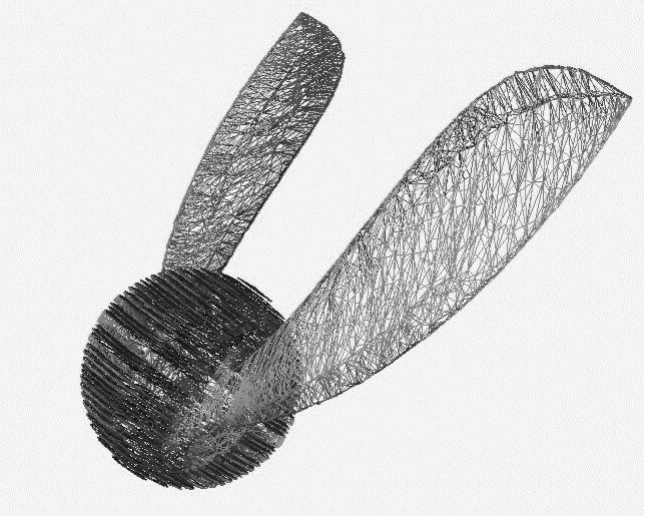

Рисунок 2: Триангулированный объект.

7. РЕЗУЛЬТАТ

Для примера была взята «Сфера Лоренца», количество точек составляет примерно 6,5 млн. точек, в итоге мы получаем около 40 000 полигонов. На данном этапе развития компьютерной техники не возможно интерактивно манипулировать миллионами точек в отличие от десятков тысяч полигонов. Таким образом, значительно уменьшился не только объем данных без потерь качества, но и требования к аппаратным ресурсам.

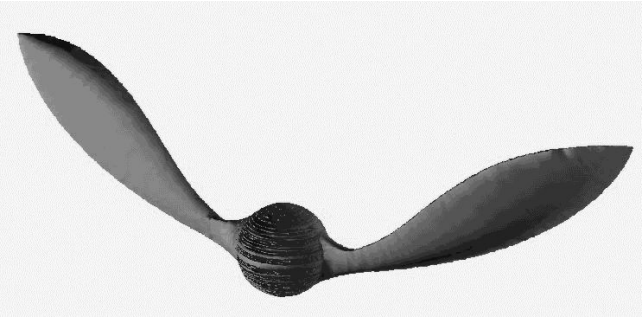

Рисунок 3: Пример окончательного результата.

8. ТЕСТИРОВАНИЕ

Помещение в куб и предобработка проводилась на двухпроцессорном сервере Intel Dual Xeon 2.4 GHz HT, 1Gb RAM, а тестирование проводилось на компьютере AMD Athlon 2800+, 512 Mb RAM, SVGA ASUS GeForce FX5200 128 Mb. Облако точек (исходный размер приблизительно 8 Gb) было помещено в воксельный куб размером 200x200x200, и после предобработки файл с данными стал занимать порядка 20 Mb. После полигонизации был получен файл размером приблизительно 2 Mb. Помещение в куб заняло приблизительно час, а процесс полигонизации воксельного объекта занял около получаса. Таким образом, результаты данной задачи, были получены за 2 часа.

9. ЗАКЛЮЧЕНИЕ

На базе данной системы можно создать достаточно универсальное средство для решения широкого круга задач, связанных с огромным количеством входных данных (порядка миллиардов точек) для последующей их визуализации. Причем результат можно получить в виде воксельного или полигонального представления, в зависимости от специфики задачи.