Качественная объёмная визуализация гигавоксельных массивов в блочном представлении на примерах данных из медицины

Николай И. Гаврилов, Вадим Е. Турлапов
факультет Вычислительной Математики и Кибернетики
Нижегородский государственный университет, Н.Новгород, Россия
gavrilov86@gmail.com; vadim.turlapov@cs.vmk.unn.ru

Аннотация

Обсуждаются подходы, применённые авторами для достижения производительности и качества объёмного рендеринга на GPU методом Volume Raycasting, достаточных для интерактивной визуализации гигавоксельных томограмм. Изложены особенности разбиения на блоки, порядок трассировки блоков для лучшей производительности, а также методы подавления артефактов рендеринга и повышения качества визуализации: зашумление стартовых позиций лучей, трикубическая интерполяция, прединтегрированный рендеринг, адаптивное изменение и дробление шага луча. Представлены результаты вычислительных экспериментов на видеокартах среднего и высокого класса и обсуждаются их результаты.

Keywords: *GPU-Based, Volume Raycasting, split into blocks, pre-integrated volume rendering, adaptive ray-step.*

1. ВВЕДЕНИЕ

С ростом возможностей источников данных и графических процессоров постепенно сложились следующие изменения в развитии объемной визуализации: 1)внимание авторов обратилось на гига- и мультигига-воксельные объекты; 2)основным подходом к решению стала декомпозиция исследуемого объема; 3)ведущим методом стал Volume Raycasting; 4)ведущей проблемой стала проблема обеспечения высокого качества и функциональности изображения при сохранении интерактивности; 5)меняются отношения в решении задачи между центральным и графических процессором.

Так, один из признанных авторитетов в области качества визуализации, компания Fovia, реализовала коммерческую библиотеку для объёмного рендеринга High Definition Volume Rendering® (http://www.fovia.com/), которая обеспечивает визуализацию в реальном времени для массива 4096^3 или 64 гигавокселя. Один из секретов высокого качества визуализации - адаптивное изменение шага луча (http://www.vizworld.com/2010/10/high-definition-volume-rendering-fovia-cpu/). Авторы работ [9] и [3] рассматривают задачи с массивами данных объемом в 8192^3 и даже 16384^3 вокселей соответственно.

В случае использования GPU идеальным хранилищем объёмных данных является трёхмерная текстура, доступная при использовании расширений библиотеки OpenGL. Однако, существуют ограничения на размеры текстуры: 512^3 вокселей. Использование блочного представления данных позволяет обойти это ограничение, хотя в целом проблема ограниченности памяти, доступной GPU, конечно остаётся. Разумеется, что CPU тоже будет успешно оперировать лишь с объемом данных, помещающимся в оперативную память. Поэтому идея блочного представления в том или ином виде используется практически во всех работах последних лет и максимально плодотворна и на CPU. Кроме декомпозиции на прямоугольные блоки, используются также блоки с полигональными границами [4]: даже невыпуклая полигональная граница способна повысить производительность вычислений. Есть попытки провести декомпозицию одновременно объема и окна визуализации согласованную с направлением лучей, получившее название slab-based рендеринга [8], чтобы уменьшить кэш-промахи для блоков, лежащих вдоль пакета лучей, находящегося в разделяемой памяти, стартующего с малого фрагмента окна. Показано также и непостоянство выигрыша и его ограниченная амплитуда (порядка 30%) и, в конечном счете, сложность лавирования между плюсами и минусами метода. В ряде работ используются октодеревья [3],[6]: в работе [3] - с целью организовать уровни разного разрешения (подобно [1]) для гигавоксельного объема, а в работе [6] - для организации сложной полигонально-воксельной сцены.

Метод Volume Raycasting показал себя как успешный инструмент и удаления пустого пространства на пути луча [13], и получения за один проход сразу несколько полезных для диагностики типов визуализации [2], и в визуализации данных смешанного разрешения [1].

Проблема обеспечения высокого качества и функциональности изображения при сохранении интерактивности включает в себя два направления: 1)обеспечение качества визуализации за счет минимизации шумов и артефактов, в том числе за счет различных модификаций метода *pre-integrated volume rendering* [11] и сплайн-интерполяции исходных данных; 2)обеспечение многообразного и качественного освещения реконструируемого объекта, обсуждаемого в работах [10], [7], [5]. Очередной виток интереса к освещению в 2011-2012 не случаен: он следствие того, что каждый из перечисленных выше методов вносит дополнительную сложность в реализацию для него полноценного освещения трехмерной сцены, отнимает заметный кусок интерактивности, требует дополнительной оптимизации вычислений.

2. МЕТОДЫ И АЛГОРИТМЫ

В нашей работе также используется Volume Raycasting и блочная декомпозиция данных, как единственная и эффективная возможность работы с гигавоксельными данными. При декомпозиции блоки данных записываются в разные трёхмерные текстуры. Текстуры имеют одинаковые размеры, за исключением текстур пограничных блоков. Чтобы избежать артефактов в местах соединения, соседние блоки должны перекрываться на толщину как минимум в один воксель (рис.1). Этого будет достаточно, если в алгоритме рендеринга при выборках значений из данных используется только трилинейная интерполяция. В нашей реализации блоки перекрываются на три вокселя, поскольку, во-первых, для реализации локального

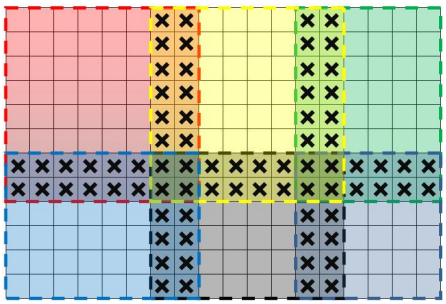


Рис.1: Иллюстрация декомпозиции данных на блоки.

освещения по Фонгу вычисляем градиент, и, во-вторых, мы используем трикубическую интерполяцию выборки вместо трилинейной, в том числе и при вычислении градиента.

2.1 Плюсы и минусы декомпозиции

Плюсы:

1) Возможность загружать бо́льшие массивы данных, ограниченные вместимостью видеокарты (при 1 Гб можно визуализировать массив данных размером 512x512x2048).

2) Декомпозиция значительно повышает производительность визуализации, поскольку при рендеринге блока выборка идёт из текстуры малого размера, что значительно быстрее выборки из текстуры большого размера. Блоки выводятся в порядке от наблюдателя: часть блоков экранируется пройденными ранее, что вызывает раннее завершение луча.

3) Пространство трассировки квантовано размерами блоков. Стратегия пропуска «пустых» областей реализуема еще производительнее. Например, для КТ томограмм обычно можно отбросить порядка 40% блоков, как не содержащих полезной информации.

Минусы:

1) Более высокая сложность алгоритма рендеринга – главный недостаток блочного представления. При рендеринге ничего не известно о данных из соседних блоков за исключением слоя перекрытия с соседними блоками. Усложняется реализация отбрасывания теней, различных нелокальных техник освещения, и вообще всех техник, требующих генерирования вторичных лучей. Также усложняется рендеринг при использовании алгоритмов, в которых луч может нести некую добавочную информацию: в технике MIDA [2] для каждого луча, кроме накопленных цвета и непрозрачности, хранится максимальная интенсивность.

2) Усложнение алгоритма мульти-объёмного рендеринга, при котором нужно производить совместный рендеринг двух или более перекрывающихся в пространстве массивов объёмных данных, каждый из которых имеет блочное представление.

3) Перекрытие блоков означает, что приграничные воксели будут продублированы, поэтому при слишком мелком разбиении (в эксперименте - при размере блоков меньше 32^3) экономия памяти GPU сведётся на нет.

4) При слишком мелком разбиении на блоки заметно увеличивается время, затрачиваемое на “переключение” между блоками. Результат рендеринга очередного блока необходимо скопировать из текстуры, в которую производился рендеринг, в текстуру, из которой будет читаться достигнутые значения (производительность значительно падает уже при размере регулярного блока 32^3 вокселей).

2.2 Пропуск пустых областей

Метод пропуска пустых областей применяется наиболее часто в CPU реализациях объёмного рендеринга. Используются оптимизирующие структуры: октодеревья, kd-деревья, полигональные сетки, ограничивающие трассируемый объём. В нашем случае для любой позиции камеры блок задаётся ограничивающим параллелепипедом, который определяет начальные и конечные точки движения лучей внутри него. Размер пограничных блоков минимизируется под размер наблюдаемых данных (рис.2).

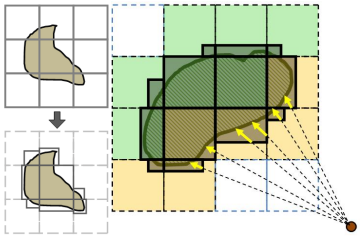


Рис 2: Подгонка ограничивающих параллелепипедов к областям данных.

Подгонка производится автоматически при любом изменении точки наблюдения и передаточной функции (transfer function). Для реализации раннего завершения луча при каждом изменении точки наблюдения для блоков производится разметка порядка трассировки (рис.3).

Результат трассировки очередного блока передается в следующий блок через текстуру. Библиотека OpenGL обеспечивает средства для рендеринга в текстуру через экранный буфер (Frame Buffer Objects). Используются две текстуры, в одну из которых производится рендеринг блока (текстура для записи), а другая хранит результат рендеринга предыдущих блоков (текстура для чтения).

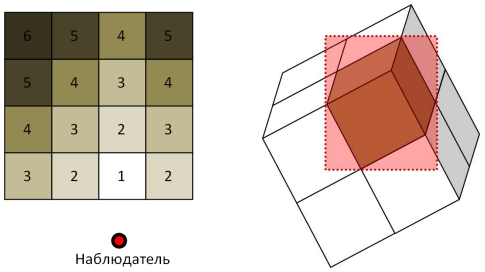


Рис. 3: Очерёдность вывода блоков (слева) и область экрана, копируемая в текстуру для чтения после рендеринга очередного блока данных (справа).

Время переключения между блоками в основном определяется временем копирования из текстуры для записи в текстуру для чтения. Для сокращения времени этого копирования мы копируем только ту часть экрана, в пределах которой производился рендеринг (рис.3).

2.3 Повышение качества визуализации

2.3.1 Подавление артефактов DVR

В методе бросания лучей луч движется с неким шагом через объём, делая на каждом шаге выборку значения из данных с использованием какой-либо интерполяции. Текущему значению данных передаточная функция ставит в соответствие оптические свойства, как правило, цвет и прозрачность, которые вносят вклад в искомый цвет

пикселя: луч пошагово “накапливает” цвет и непрозрачность. Благодаря возможной нелинейности передаточной функции, на интервале между двумя соседними отсчетами данных, цветовая интерпретация данных интерполированных передаточной функцией может существенно отличаться по оптическим свойствам от точек на концах отрезка. Это приводит к артефактам постклассификации. Из-за регулярности расположения стартовых позиций лучей эти артефакты имеют вид концентрических замкнутых линий (рис.4). Один из реализованных методов борьбы с данным артефактом - сдвиг стартовых позиций лучей на сонаправленный случайный вектор: накопление зашумлённых изображений даст усреднённое изображение без шума. Накопление кадров целесообразно применять при отсутствии мощного графического процессора, который был бы способен произвести рендеринг в реальном времени с достаточно малым шагом луча. Обычно для данных томографии при шаге менее 1/8 от размера вокселя артефакты уже не видны (рис.6).

Для устранения артефактов, связанных с недостаточной гладкостью исходных данных (в виде “ступенек”), реализована трикубическая интерполяция исходных данных «налету». Для этого опубликованный алгоритм бикубической фильтрации [12] расширен на трёхмерный случай. Для выполнения одной выборки с такой фильтрацией необходимо выполнить 8 выборок с трилинейной фильтрацией (см. рис.4).

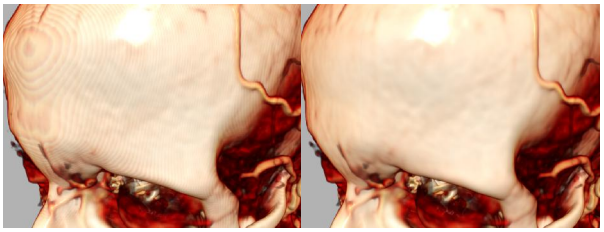


Рис. 4: Сравнение результатов рендеринга КТ томограммы при использовании обычной трилинейной интерполяции (слева) и трикубической интерполяции (справа).

2.3.2 Адаптивное изменение шага луча

Несмотря на возможность компенсировать недостаточно большой шаг луча техникой накопления кадров, более правильным подходом для рендеринга на видеокартах высокого класса является рендеринг одного изображения высокого качества, т.е. с малым шагом луча – объёмный рендеринг высокого разрешения (High definition volume rendering). Современные графические процессоры могут обеспечивать визуализацию в реальном времени больших данных с частотой выборки 8 на один воксель. Однако, производительность такого рендеринга можно значительно повысить, изменяя длину шага луча адаптивно в зависимости от значения данных в текущей позиции, не ухудшая при этом качества рендеринга.

Предлагаемый нами алгоритм иллюстрирован на Рисунке 5. Мы трассируем луч с неким большим шагом (0,5 от размера вокселя) и на каждом шаге делаем выборку из данных в текущей позиции луча. В случае если мы попали в область видимых данных (т.е. значение текущей выборки не соответствует полной прозрачности), то мы сдвигаем наш луч на шаг назад и начинаем идти с шагом 1/8 от стандартного шага. При этом делается 8 шагов и накапливается цвет и непрозрачность так же, как в обычном рендеринге с постоянным шагом луча. На видеокарте GeForce GTX 580 при шаге луча 1/8 такая оптимизация ускоряет рендеринг в среднем в 3 раза, при шаге 1/16 – в 6.2 раза.

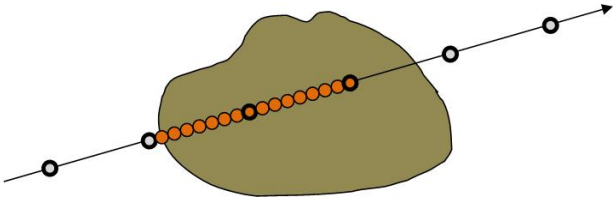


Рис. 5: Иллюстрация дробления шага луча в зависимости от значения выборки из данных. Луч двигается с большим шагом (жирные точки) и в случае нахождения в видимой области производится серия из 8 выборок на отрезке от предыдущего шага до текущего (восьмая выборка совпадает с позицией текущего шага).

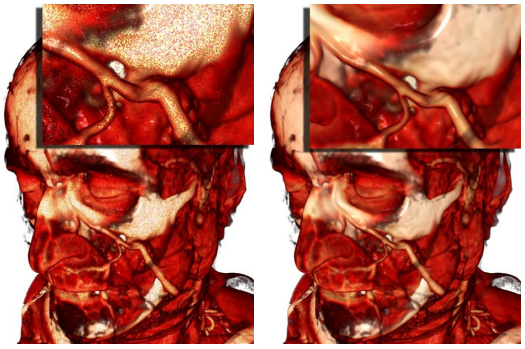


Рис. 6: Сравнение качества рендеринга (слева направо) при шаге луча в 1 и 1/8 от размера вокселя.

2.3.3 Дробление шага луча без увеличения количества выборок из данных

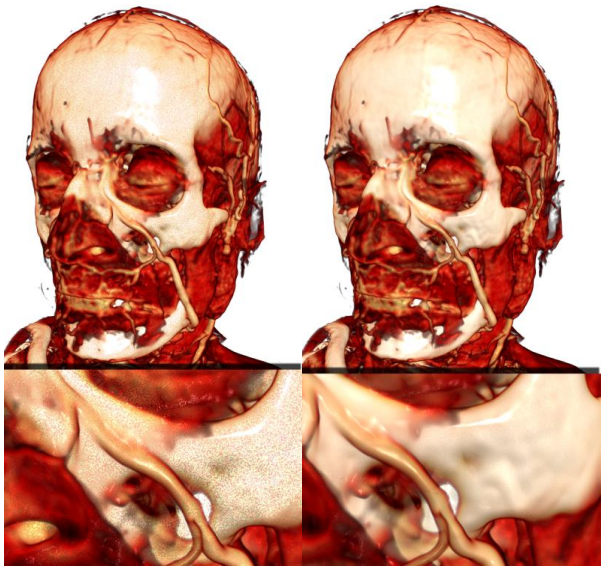


Рис. 7: Сравнение качества обычного (а) (4.6 fps) и метода разбиения шага луча (б) (4.2 fps). Шаг луча – 0,5 от размера вокселя.

Уменьшение шага луча хорошо устраняет артефакты, но число выборок при этом значительно снижает производительность. Предложен и исследован *метод дробления шага* луча без увеличения количества выборок

из данных. Шаг луча уменьшен в 20 раз, но выборка из данных выполняется только каждые 20 шагов. Внутри шага выборки используются интерполированные значения. Уже линейная интерполяцию даёт значительное улучшение качества (рис.7).

2.3.4 *Прединтегрированный объёмный рендеринг (Pre-Integrated Volume Rendering)*

Довольно эффективным методом устранения артефактов, связанных с “пролётом” луча мимо видимых особенностей в пространстве, является прединтегрированный объёмный рендеринг [11]. Идея объёмного рендеринга состоит в вычислении интеграла рендеринга вдоль луча для каждого пикселя экрана методом прямоугольников. Идея метода прединтегрированного рендеринга состоит в замене метода прямоугольников на метод более высокого порядка. Уже метод трапеций даёт значительное улучшение качества. Прединтегрированный рендеринг особенно хорошо (рис.8) сказывается на визуализации тонких слоев, которые могут быть пропущены в обычном рендеринге (избавиться от этого иногда не помогает и зашумление позиции старта луча - артефакты остаются даже при усреднении по десяткам кадров).

Данный метод отличается от метода разбиения шага луча тем, что здесь мы стараемся точно вычислить интеграл на отрезке между соседними выборками на луче (рис.9). В данном методе используется прединтегрированная таблица, которая для любых двух значений данных *(a, b)* хранит цвет и непрозрачность, которые должен накопить луч, при проходе по отрезку, в начальной точке которого выборка из данных была равна *a*, в конечной – *b*. При изменении передаточной функции таблица пересчитывается.

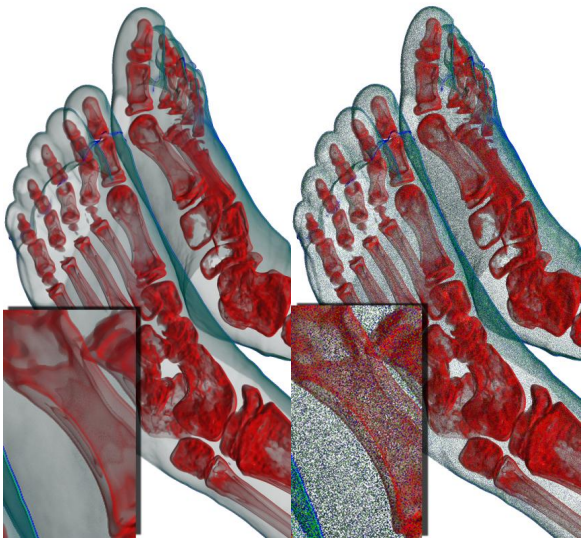


Рис. 8: Сравнение качества обычного (справа) и прединтегрированного рендеринга (слева). Шаг луча – 0,5 от размера вокселя.

Недостаток прединтегрированного рендеринга – необходимость вычислять прединтегрированную таблицу всякий раз при изменении передаточной функции. На практике изменение передаточной функции происходит в процессе интерактивной визуализации постоянно. Причём, в подходах, использующих методы интегрирования более высоких порядков размерность таблицы увеличивается. Так, для метода парабол нужна трёхмерная таблица, которая будет хранить оптические свойства отрезков путей луча, которые будут определяться уже тремя значениями [11].

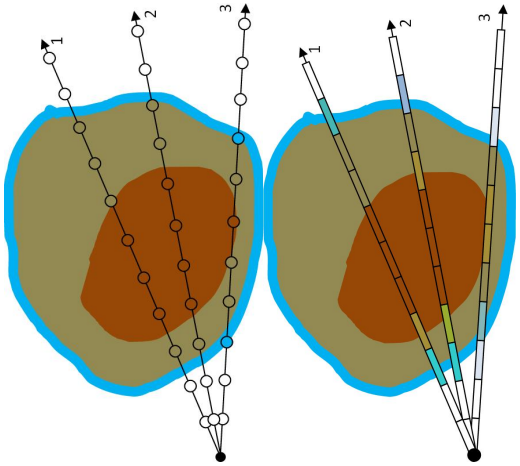


Рис. 9: Иллюстрация обычного рендеринга, при котором часть лучей не попадают в синюю область (слева) и прединтегрированного рендеринга (справа).

2.3.5 *Артефакты на границах блоков*

Как было описано выше, для устранения видимых швов на границах соседние блоки должны перекрываться на границах, в нашей реализации толщина перекрытия соседних блоков составляет три вокселя, что позволяет правильно вычислять на границе блока градиент, даже при использовании трикубической интерполяции. Однако, для устранения артефактов также необходимо, чтобы для каждого пикселя искомого изображения путь луча был непрерывен. При рендеринге очередного блока лучи должны стартовать в позициях, бывшими “следующими” для соответствующих лучей при рендеринге предыдущих блоков. Единственная доступная информация о луче-предшественнике – это координаты соответствующего пикселя и накопленные цвет и прозрачность соответствующего пикселя. Координаты пикселя используются для генерирования случайного числа, которое используется при случайном сдвиге стартовой позиции луча для подавления артефактов. Т.к. для лучей из разных блоков, но находящихся на одной линии, координаты пикселя одни и те же, то и случайный сдвиг для них будет неизменен, что позволит продолжать движение луча без разрывов.

3. ВЫЧИСЛИТЕЛЬНЫЕ ЭКСПЕРИМЕНТЫ

3.1 Производительность для видеокарт потребительского класса

Ниже представлены результаты экспериментов по замеру производительности рендеринга при выборе различных размеров блоков, на которые мы делим массив данных. В нашем случае используются кубические блоки размерами стороны 32, 48, 64, 96, 128 и 256. Для выбранных тестовых массивов данных и видеокарты GeForce GTS 250 оптимальными размерами оказались 64 и 128 в зависимости от данных. Использовано 5 тестовых массивов данных, которые общедоступны в сети. Кроме того, мы используем три различных техники визуализации:

1) Объёмный рендеринг без затенения;

2) Объёмный рендеринг с затенением по локальной модели освещения Фонга;

3) Объёмный рендеринг без затенения с трикубической выборкой вместо трилинейной.

В данных экспериментах мы перекрываем наши блоки на толщину в два вокселя. Размеры окна вывода: 800x600 пикселей. Шаг луча в алгоритме Ray Casting: 0,34 от длины диагонали вокселя. Использованная видеокарта: GeForce GTS 250.

Рис. 10: Тестовые данные: A) hand 244x124x257 (КТ данные); B) Beetle 832x832x494; C) melanix 512x512x1203 (КТ данные); D) x-mas 512x512x999 (КТ ёлки); E) vessels 512³ (МРТ томограмма сосудов головного мозга).

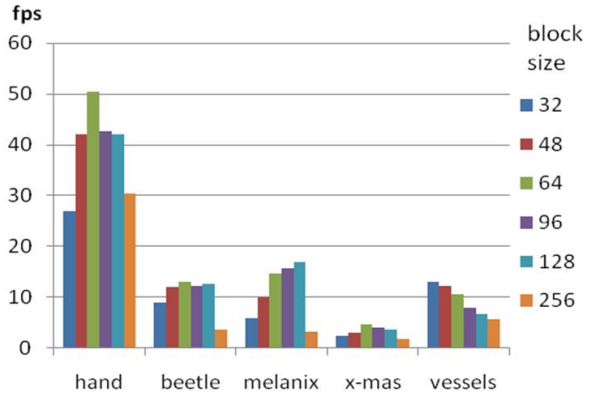


Диаграмма 1: Средняя по различным методам рендеринга производительность рендеринга при различных размерах блока разбиения данных.

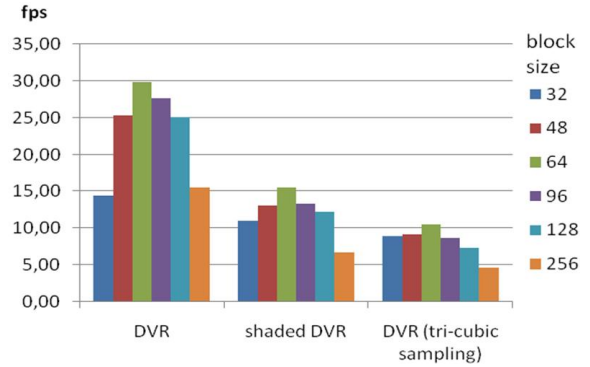


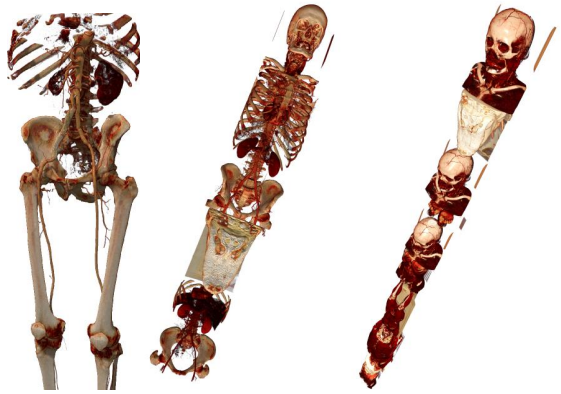
Диаграмма 2: Средняя по тестовым данным производительность рендеринга при различных размерах блока разбиения.

3.2 Ускорение оптимизированных подходов для видеокарты высокого класса

Для следующих экспериментов использовалась видеокарта GeForce GTX 580, размер блока данных был выбран равным 256³. При более мелком разбиении количество блоков становиться довольно большим, что сильно увеличивает суммарное время на переключение между блоками, особенно при их большом количестве. Так, массив данных multi2 был разбит на 88 блоков размером 256³ вокселей каждый, не считая блоков меньших размеров на границах массива. Учитывая, что данные хранились на GPU в 12-битном формате (типичном для медицинских данных), весь массив занял 2Гб памяти GPU .

Для получения тестовых данных больших размеров в один массив были искусственно объединены различные томограммы. На диаграммах 3-4 представлены результаты замеров производительности рендеринга с переменным шагом луча и ускорение производительности от этой оптимизации.

В данных экспериментах мы перекрываем наши блоки на толщину в три вокселя, т.к. помимо трёх режимов рендеринга, использованных ранее, мы использовали режим рендеринга с локальным освещением с трикубической интерполяцией (режим cubic sDVR на диаграммах 3-4). Размеры окна вывода: 1920x1018 пикселей. Шаг луча в алгоритме Ray Casting: 1/8 от длины стороны вокселя. Использованная видеокарта: GeForce GTX 580 3Gb.



(а) (б) (в)

Рис. 11: Тестовые данные: а) amnesix 512x512x1624; б) multi1 512x512x1981; в) multi2 512x512x5382.

Благодаря технике изменения шага луча производительность рендеринга гораздо менее чувствительна к уменьшению длины шага луча. Дело в том, что регулярный шаг луча мы оставляем неизменным, но мы увеличиваем в два раза число выборок только в областях пространства, где данные видимы. Тестовые данные и настройки визуализации таковы, что луч обычно очень скоро накапливает непрозрачность и останавливается.

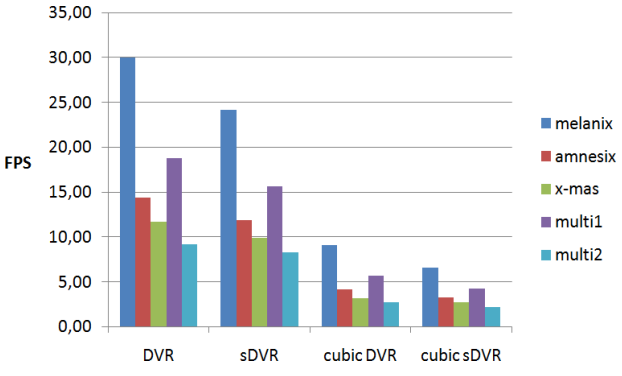


Диаграмма 3: Производительность рендеринга с адаптивным шагом луча

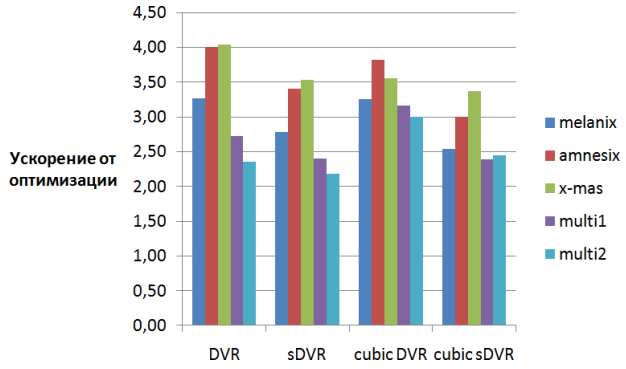


Диаграмма 4: Ускорение рендеринга с адаптивным изменением шага луча по сравнению с неоптимизированной версией при тех же условиях.

Однако, для случая среды с низкой непрозрачностью, в которой луч может трассироваться большое число шагов и на каждом из них накапливать цвет и непрозрачность, понадобится другой способ подстройки шага, при котором длина шага будет меняться более плавно при изменении оптических свойств среды.

Массив данных x-mas отличен от остальных тем, что имеет сложную топологию видимой области пространства, что затрудняет использование регулярных ускоряющих структур и ограничивающей геометрии для пропуска пустых областей. Техника изменения шага луча не восприимчива к топологии данных и является достаточно универсальной.

4. ЗАКЛЮЧЕНИЕ

Рассмотрены современные тенденции в развитии объемной визуализации и детали комплексного решения для интерактивного визуализатора гигавоксельных томограмм. Основу составила межплатформенная реализация на GPU блочной декомпозиции гигавоксельных данных для метода Volume Raycasting. Изложены особенности разбиения на блоки в условиях моделирования освещения реконструируемого объекта, устранение нежелательных артефактов, порядок рендеринга блоков для лучшей производительности, методы подавления артефактов рендеринга и повышения качества визуализации: трикубическая интерполяция, реализация прединтегрированного рендеринга, методы дробления и адаптивного изменения шага луча.

Результаты экспериментов показали высокую производительность рендеринга для видеокарты GeForce GTX 580 3Gb, достаточную для визуализации гигапиксельных томограмм.

Благодаря методам устранения артефактов и увеличения частоты выборок при рендеринге качество интерактивной визуализации сравнимо с качеством CPU-реализации от компании Fovia.

С ростом объемов данных и усложнением сцен продолжит расти значение оптимизирующих структур и методов, как и массивно-параллельных устройств, обеспечивающих потоковые вычисления нижнего уровня. Следует ожидать повышения роли одновременно и CPU, и GPU, и, особенно, их совместного применения.

5. ЛИТЕРАТУРА

[1] J.Beyer et al,2008, *Smooth Mixed-Resolution GPU Volume Rendering*, IEEE International Symposium on Volume and Point-Based Graphics (VG ’08); 2008. pp. 163–170.

[2] S.Bruckner et al, 2009, *Instant volume visualization using maximum intensity difference accumulation*. Computer Graphics Forum, 28(3):775–782, June 2009.

[3] C.Crassin et al, 2009, *GigaVoxels: Ray-Guided Streaming for Efficient and Detailed Voxel Renderin*. ACM SIGGRAPH Symp. on Interactive 3D Graphics and Games (I3D) - feb 2009, 8p.

[4] B.Kainz et al, 2009. *Ray Casting of Multiple Volumetric Datasets with Polyhedral Boundaries on Manycore GPUs*, Proceedings of ACM SIGGRAPH Asia 2009, Volume 28, No 152.

[5] J.Kronander et al, 2012, *Efficient Visibility Encoding for Dynamic Illumination in Direct Volume Rendering*. IEEE TVCG, Volume 18, Number 3, 2012, pp. 447-462

[6] S.Laine, T.Karras (NVIDIA), *Efficient Sparse Voxel Octrees*, IEEE Transactions on Visualization and Computer Graphics, vol. 17(8), 2011, pp. 1048-1059.

[7] F.Lindemann, T.Ropinski, 2011, *About the Influence of Illumination Models on Image Comprehension in Direct Volume Rendering*. IEEE TVCG. V.17(12), Dec.2011, pp. 1922-1931

[8] J.Mensmann, et al, 2011, *Slab-Based Raycasting*: Exploiting GPU Computing for Volume Visualization, Springer, Communications in Computer and Information Science (CCIS), V. 229. - 2011. pp. 246-259.

[9] E.Gobbetti et al, 2008, *A single-pass GPU ray casting framework for interactive out-of-core rendering of massive volumetric datasets*. - 8p.

[10] M.Hadwiger. et al, 2006, *GPU-Accelerated Deep Shadow Maps for Direct Volume Rendering*, Proc.of the 21st ACM SIGGRAPH/EUROGRAPHICS symposium on Graphics hardware, Sep.03-04, 2006, Vienna, Austria, pp. 49-52

[11] J.E.Hajjar et al, 2008, *Second order pre-integrated volume rendering* // IEEE Pacific Visualization Symposium, March 2008, pages: 9 – 16.

[12] D.Ruijters et al, 2008. *Efficient GPU-Based Texture Interpolation using Uniform B-Splines*, In IEEE Transactions on Journal of Graphics, GPU, & Game Tools, Vol. 13, No. 4, pp 61-69.

[13] V.Vidal(Inria) et al, 2008, *Simple Empty-Space Removal for Interactive Volume Rendering*, Journal of Graphics, GPU, and Game Tools, Volume 13, Issue 2, pp. 21-36