Исследование и развитие метода декомпозиции для анализа больших пространственных данных [1]

Золотов В.А., Семенов В.А.

Аннотация. Статья посвящена развитию метода декомпозиции для индексации, поиска и анализа больших пространственных данных. Главное внимание уделяется алгоритмам, основанным на регулярных октальных деревьях и обеспечивающим эффективное решение ряда вычислительных задач. Исследуемые алгоритмы определения столкновений и выборки по заданной области, в частности, применимы для моделирования сложных динамических пространственно-трехмерных сцен с объектами, имеющими протяженные границы. Для модельного набора данных на основе вероятностного анализа выводятся оценки сложности, которые обобщают и улучшают известные результаты, а также служат теоретическим обоснованием для применения алгоритмов к более широкому классу приложений.

Ключевые слова: Пространственные индексы, октальные деревья, вычислительная сложность.

1. Введение

Стремительный рост объемов информации, а также необходимость ее анализа приводят к развитию новых подходов к управлению данными и, в частности, методов пространственного индексирования, без которых невозможен быстрый поиск и обработка в геоинформационных базах данных, системах логистического обеспечения, системах автоматизации проектирования, системах управления проектами. Как правило, популярные универсальные и специализированные СУБД предусматривают для этих целей средства пространственного индексирования и поиска. Подобные средства успешно справляются с обработкой статической информации, однако часто не приспособлены для данных, подлежащих перманентным изменениям [1]. Проблемы эффективного поиска и анализа еще более усложняются, когда информация представляет собой не просто массивы точек, а сложно-структурированные наборы данных, например, множества объектов с протяженными пространственными границами [2]. Класс подобных

[1] Исследование поддержано грантом РФФИ 13-07-00294.

приложений чрезвычайно широк и охватывает не только перечисленные выше прикладные области, но и многочисленные системы компьютерной графики, визуализации и анимации. Эти факторы определяют актуальность темы и огромный интерес, как со стороны научного сообщества, так и производителей системного и прикладного программного обеспечения. В частности, активные разработки в этой области ведут компании Google, Oracle, IBM, Autodesk, Bentley, Intergraph, AVEVA, сталкивающиеся с проблемой увеличения объемов анализируемой пространственно-временной информации.

В работе [3] были системно проанализированы фундаментальные семейства методов индексации и поиска многомерных данных. В частности, рассматривались структуры поиска на интервалах, сбалансированные ветвистые деревья во внешней памяти, бинарные деревья пространственной декомпозиции, префиксные деревья, нерегулярные и регулярные многоуровневые сетки, метрические деревья, а также связанные с ними разнообразные методики хэширования, расщепления и кластеризации.

Большое внимание было уделено методам пространственной индексации на основе регулярных октальных деревьев. Главным их достоинством является простота развертывания и модификации индексов, обусловленная априори известным положением секущих плоскостей. Это позволяет относительно легко обновлять индексы и обеспечивать эффективность исполнения типовых пространственных запросов при перманентных изменениях самих данных. Однако данные методы не обеспечивают сбалансированность структур индексов и могут приводить к их деградации в тех случаях, когда данные неравномерно распределены по пространству или имеют протяженные границы. Известные оценки сложности в подобных наихудших случаях приводят к пессимистическим выводам и не определяют реальных границ применимости методов декомпозиции на основе регулярных октальных деревьев. По-видимому, они нуждаются в более детальном исследовании и развитии, исходя из оценок сложности в среднем, которые могут быть получены на основе вероятностного анализа основных алгоритмов при необходимой конкретизации условий прикладных задач.

В разделе 2 рассматривается постановка задачи, связанной с визуальным анализом пространственно-трехмерных сцен. В рамках данной постановки исследуется алгоритм развертывания регулярных октальных деревьев. В разделе 3 описываются алгоритмы определения столкновений и выборки объектов по заданной области, основанные на регулярных октальных деревьях. Полученные для них оценки сложности в среднем обобщают и существенно улучшают известные результаты, а выработанные рекомендации создают предпосылки к широкому практическому применению методов декомпозиции для индексации, поиска и анализа больших пространственных данных. В заключении подводятся итоги проведенного исследования.

2. Пространственная декомпозиция на основе октальных деревьев

Начнем с анализа классического метода пространственной декомпозиции, основанного на регулярных октальных деревьях. В трехмерном случае параллелепипед, пространственно ограничивающий весь набор данных (или в англоязычной литературе AABB — Axis Aligned Bounding Box), разбивается на восемь равных частей плоскостями, перпендикулярными каждой из координатных осей. Процесс рекурсивно применяется до тех пор, пока количество элементов данных в каждом вновь образованном октанте не окажется ниже некоторого предустановленного порога $m > 1$. С результирующим пространственным разбиением ассоциируется соответствующее дерево. Корень дерева соответствует исходному параллелепипеду, содержащему в себе весь набор данных, а вершины — вложенным октантам, группирующим данные на разных иерархических уровнях пространственной декомпозиции. Пример декомпозиции и полученного октального дерева приведен на рис. 1.

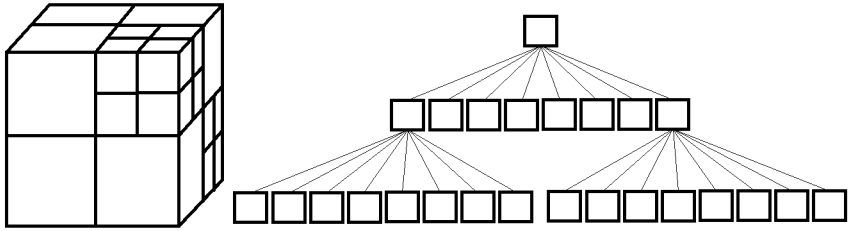


Рисунок 1. Пример пространственной декомпозиции и соответствующего ей октального дерева.

Заметим, что способ ассоциирования элементов данных с вершинами октального дерева, в конечном счете, определяется способом их дальнейшей локализации и поэтому зависит от их пространственной семантики. Точечные данные могут быть непосредственно ассоциированы с листовыми октантами. Однако геометрические объекты с протяженными границами могут занимать определенный объем исходного параллелепипеда. Поэтому более рациональным представляется их ассоциирование с теми октантами, в которых они могут быть размещены полностью [4]. Для такого способа есть и иные причины, связанные с возможным пересечением секущими плоскостями даже небольших объектов и невозможностью установить простое соответствие между ними и листовыми октантами.

Рассмотрим псевдокод алгоритма построения октального дерева, приведенный ниже:

```
FUNCTION BUILD_OCTREE(box, objects, m)
BOX box
OBJECT COLLECTION objects
```



```
INTEGER m
RETURNS POINTER NODE root_octant

root_octant = new NODE(box);

FOR_EACH (POINTER OBJECT o IN objects)
        INSERT(o, root_octant)


PROCEDURE INSERT(object,octant)
POINTER OBJECT object
POINTER NODE octant

POINTER NODE child = LOCALIZE(object, octant)
IF(child)
        RETURN INSERT(object,child)
ELSE
        ADD(object,GET_OBJECTS(octant))

IF(CAN_SPLIT(octant))
        SPLIT(octant)


PROCEDURE SPLIT(octant)
POINTER NODE octant

DECOMPOSE(octant)
FOR_EACH(OBJECT POINTER object IN GET_OBJECTS(octant))
        {
        POINTER NODE child = LOCALIZE(object,octant)
        if(child)
                {
                ERASE(object,GET_OBJECTS(octant))
                INSERT(object, child)
                }
        }
```

Алгоритм 1. Построение регулярного октального дерева.

Функция BUILD_OCTREE принимает в качестве входных параметров ограничивающий параллелепипед box, множество объектов (элементов данных с пространственной семантикой) $objects$, а также параметр заполнения вершин октального дерева m. Возвращаемое значение — указатель на корень развернутого октального дерева. Для построения используется вспомогательная процедура INSERT, которая выполняет вставку

заданного объекта $object \in objects$ в октальное дерево, начиная с заданного родительского октанта $octant$. Остановимся на этой процедуре более подробно. В псевдокоде процедуры используется вспомогательная функция LOCALIZE, которая определяет, в каком из дочерних октантов может быть полностью размещен объект. Если исходный октант является листовым или объект не локализуется ни в одном из дочерних октантов, функция возвращает нулевое значение. При успешной локализации процедура вставки рекурсивно повторяется. В противном случае, посредством процедуры ADD, объект приписывается исходному октанту (более точно, коллекции объектов, получаемой при помощи вспомогательной функции GET_OBJECTS). Функция CAN_SPLIT проверяет условие наполнения октанта и целесообразность его разбиения, которое реализуется при помощи процедуры SPLIT. При вызове процедуры DECOMPOSE создается восемь дочерних октантов и предпринимается попытка перераспределить по ним объекты родительского октанта. Перенос объектов в дочерние октанты осуществляется посредством процедур INSERT и ERASE.

Определение. Октальное дерево, построенное на основе рассмотренного алгоритма 1 с порогом наполнения октантов m, назовем регулярным и обозначим $Octree(m)$.

Регулярное октальное дерево может использоваться для решения различного рода вычислительных задач, связанных с поиском и анализом пространственно-временных данных. Естественно, что затраты на развертывание дерева должны сполна компенсироваться более быстрым решением целевых задач. Оценим затраты на построение октального дерева. Заметим, что подобные оценки могут существенно зависеть от специфики прикладных данных и анализ наихудшего случая, как правило, приводит к довольно пессимистическим результатам, не отражающим реальные показатели производительности для большинства приложений. Поэтому получим оценки сложности в среднем на основе вероятностного анализа некоторого упрощенного набора данных.

Для этого рассмотрим набор данных, связанный с визуальным анализом пространственно-трехмерных сцен и допускающий содержательную параметризацию. Пусть сцена представляется набором n трехмерных геометрических объектов, равномерно расположенных внутри единичного куба. Ограничивающим объемом каждого объекта является куб с характерным размером сторон $0 \le l \le 1$. В рамках подобной постановки x, y, z-координаты центров являются независимыми равномерно распределенными величинами на соответствующих отрезках $[l, 1-l]$. Для обсуждаемых задач анализа сцен будет применяться единая техника пространственной локализации объектов на основе их ограничивающих объемов, поэтому в дальнейшем мы не делаем никаких различий между понятиями геометрического объекта и его ограничивающего параллелепипеда. Рисунок 2а иллюстрирует область возможного расположения центров объектов внутри объема всей сцены. При

этом величина $\rho(n,l) = \frac{nl^3}{(1-l)^3}$ определяет эффективную пространственную плотность данных.

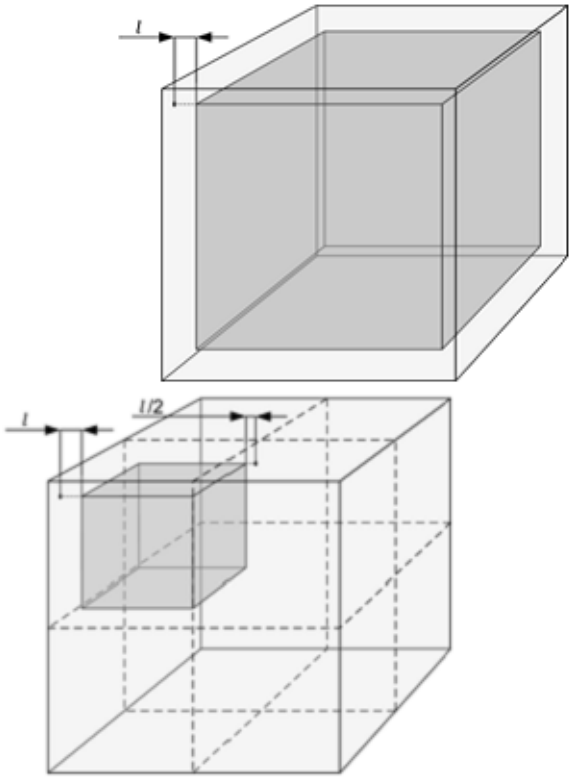


Рисунок 2. а) Область возможного расположения центров объектов внутри ограничивающего объема набора данных. б) Область возможного расположения центров объектов, локализуемых в октантах следующего уровня.

Определение. Набор n кубов назовем модельным и обозначим как $S(n,l)$, если они равномерно распределены в единичном кубе, а их ребра ориентированы вдоль главных координатных осей и имеют относительный размер $0 \le l < \frac{1}{2}$.

Заметим, что при относительном размере объектов $l \ge \frac{1}{2}$ октальная структура вырождается, поскольку все объекты локализуются в самом верхнем октанте при глубине дерева $h = 1$. Поэтому при дальнейшем рассмотрении мы ограничимся содержательным случаем $l < \frac{1}{2}$. Исследуем свойства регулярных октальных деревьев, развернутых для модельного набора данных. Для этого сформулируем и докажем несколько вспомогательных утверждений.

Лемма 1. Пусть регулярное октальное дерево глубиной $h \le \left\lceil \log_2 \frac{1}{l} \right\rceil$ построено для модельного набора $S(n,l)$. Тогда математическое ожидание числа объектов в октантах *i*-го уровня и ниже ($1 \le i \le h$) есть

$$n_i = n\frac{(1-2^{i-1}l)^3}{(1-l)^3}$$

Доказательство:

Прежде всего, заметим, что глубина регулярного октального дерева всегда ограничена величиной $\left\lceil \log_2 \frac{1}{l} \right\rceil$, поскольку объекты модельного набора не могут быть локализованы в октантах размера равного или меньшего, чем l. Поэтому анализу подлежат лишь допустимые уровни локализации объектов $1 \le i \le \left\lceil \log_2 \frac{1}{l} \right\rceil$.

Далее, уточним область трехмерного пространства, где могут располагаться центры объектов, приписанных октантам разных уровней. На рисунке 2а темным цветом выделена область возможного расположения центров объектов в октанте верхнего уровня, поскольку они не могут находиться к граням ограничивающего параллелепипеда ближе, чем на расстоянии $\frac{l}{2}$. По построению дерева объект приписывается родительскому октанту, если он пересекается одной из секущих плоскостей. В противном случае предпринимается попытка локализовать его в одном из дочерних октантов следующего уровня. На рисунке 1б темным цветом показана одна из таких областей в октантах второго уровня. Чтобы оценить число объектов, локализуемых на втором уровне дерева или ниже, воспользуемся предположением об их равномерном пространственном распределении и геометрической интерпретацией вероятности.

Вероятность того, что объект не будет пересечен секущими верхнего октанта и, следовательно, будет локализован на втором уровне дерева или ниже, равна

$$P_2^1 = \frac{(1-2l)^3}{(1-l)^3}$$

В знаменателе выражения — объем области допустимого расположения центров объектов в октанте первого уровня, в числителе — суммарный объем областей возможного расположения центров объектов, локализуемых в восьми дочерних октантах. Рассуждая аналогично для оставшихся уровней, получим вероятность того, что объект не будет пересечен секущими *i*-го октанта и будет локализован на следующих уровнях дерева

$$P_{i+1}^i = \frac{(\frac{1}{2^{i-1}} - 2l)^3}{(\frac{1}{2^{i-1}} - l)^3} = \frac{(1-2^i l)^3}{(1-2^{i-1}l)^3}$$

Тогда математическое ожидание числа объектов, локализованных в октантах *i*-го уровня и ниже ($1 \le i \le h$) выражается как

$$n_i = n \cdot P_2^1 \cdot P_3^2 \cdot P_4^3 \cdot \ldots \cdot P_{i-1}^{i-2} \cdot P_i^{i-1}$$

или

$$n_i = n\frac{(1-2l)^3}{(1-l)^3} \cdot \frac{(1-4l)^3}{(1-2l)^3} \cdot \frac{(1-8l)^3}{(1-4l)^3} \cdot \ldots \cdot \frac{(1-2^{i-1}l)^3}{(1-2^{i-2}l)^3}$$

$$= n\frac{(1-2^{i-1}l)^3}{(1-l)^3}$$

Что и требовалось доказать.

Лемма 2. Пусть регулярное октальное дерево глубиной $h \le \left\lceil \log_2 \frac{1}{l} \right\rceil$ построено для модельного набора $S(n,l)$, тогда математическое ожидание числа объектов в октанте *i*-го уровня есть

$$N_i = \begin{cases} n\dfrac{(1-2^{i-1}l)^3 - (1-2^i l)^3}{8^{i-1}(1-l)^3}, 1 \le i < h \\ n\dfrac{(1-2^{i-1}l)^3}{8^{i-1}(1-l)^3}, i = h \end{cases}$$

Доказательство:

Лемма 1 определяет ожидаемое число объектов в октантах *i*-го уровня и ниже как n_i, а ожидаемое число объектов в октантах *i+1*-го уровня и ниже как n_{i+1}. Поскольку на *i*-ом уровне дерева имеется 8^{i-1} октантов, то математическое ожидание числа объектов в одном таком октанте определяется выражением

$$N_i = \frac{1}{8^{i-1}}(n_i - n_{i+1}) = n\frac{(1-2^{i-1}l)^3 - (1-2^i l)^3}{8^{i-1}(1-l)^3}$$

Выражение справедливо для всех октантов уровней $1 \le i < h$. Для определения ожидаемого числа объектов в листовых октантах воспользуемся непосредственно леммой 2. Тогда для октантов уровня $i = h$ имеем

$$N_i = n\frac{(1-2^{i-1}l)^3}{8^{i-1}(1-l)^3}$$

Что и требовалось доказать.

На основе доказанных лемм удается оценить глубину регулярного октального дерева. Имеет место следующая теорема.

Теорема 1. Ожидаемая глубина регулярного октального дерева $Octree(m)$, построенного для модельного набора данных $S(n,l)$ с плотностью пространственного заполнения $\rho(n,l)$, определяется следующим выражением:

$$h = \begin{cases} \left\lceil \log_2 \dfrac{2}{l + (1-l)\sqrt[3]{\dfrac{m}{n}}} \right\rceil, & m \geq \rho(n,l) \\ \left\lceil \log_2 \dfrac{1}{l} \right\rceil, & m < \rho(n,l) \end{cases}$$

Доказательство:

Согласно алгоритму построения регулярного октального дерева его глубина ограничивается следующими факторами:

— размер листового октанта должен быть строго больше габаритов объекта l, чтобы объект мог быть размещен в нем целиком;

— разбиение октантов возможно лишь при их достаточном наполнении, превышающем заданный порог m.

Чтобы процедура разбиения могла быть применена к октантам *i*-го уровня в соответствии с леммой 2, должны выполняться следующие условия:

$$\begin{cases} \dfrac{1}{2^{i-1}} > 2l \\ n\dfrac{(1-2^{i-1}l)^3}{8^{i-1}(1-l)^3} > m \end{cases}$$

В листовых октантах, по крайней мере, одно из приведенных условий не выполняется. Невыполнение первого условия приводит к неравенству $\frac{1}{2^{h-1}} \leq 2l$, а в комбинации с ранее полученной оценкой $h \leq \left\lceil \log_2 \frac{1}{l} \right\rceil$, имеем для глубины дерева строгое равенство $h_1 = \left\lceil \log_2 \frac{1}{l} \right\rceil$.

Рассмотрим более подробно второе условие. Условия разбиения должны выполняться в октантах предпоследнего уровня и не должны выполняться в листовых октантах с ожидаемым числом объектов, не превышающим заданный порог m. Следовательно, имеют место следующие неравенства:

$$n\frac{(1-2^{h-1}l)^3}{8^{h-1}(1-l)^3} \leq m < n\frac{(1-2^{h-2}l)^3}{8^{h-2}(1-l)^3}$$

Их согласованный анализ приводит к следующей оценке глубины дерева:

$$h_2 = \left\lceil \log_2 \frac{2}{l + (1-l)\sqrt[3]{\dfrac{m}{n}}} \right\rceil$$

Поскольку рост дерева прекращается, когда нарушается одно из условий разбиения октантов, то ожидаемая глубина регулярного октального дерева будет определяться минимальным из найденных значений $h = \min(h_1, h_2)$.

Сравнив найденные значения, получим соотношение параметров, при котором глубина определяется первым фактором. В самом деле, из условия $h_1 \leq h_2$ следуют соотношения $\log_2 \frac{2}{l+(1-l)\sqrt[3]{\frac{m}{n}}} \leq \log_2 \frac{1}{l}$, $l \leq (1-l)\sqrt[3]{\frac{m}{n}}$, $m \geq \frac{nl^3}{(1-l)^3}$ или $m \geq \rho(n,l)$ с учетом определения эффективной пространственной плотности данных. Тем самым, имеет место $h = \begin{cases} h_1, & m \geq \rho(n,l) \\ h_2, & m < \rho(n,l) \end{cases}$ и теорема доказана.

3. Теоретический анализ сложности метода декомпозиции

Итак, проанализируем вычислительные затраты на развертывание регулярного октального дерева, а также на выполнение запросов, связанных с выборкой объектов по заданной пространственной области и определением столкновений.

Как отмечалось, асимптотические оценки сложности, а также оценки сложности в наихудшем случае не отражают реальную эффективность метода декомпозиции для многих важных классов приложений. Поэтому воспользуемся описанным модельным набором данных и получим оценку сложности на основе проведенного вероятностного анализа и полученных результатов о свойствах октальных деревьев. Существенной стороной проводимого анализа является попытка получить оценки не в асимптотическом приближении, а на всем диапазоне изменения размерности и других параметров задачи. При анализе сложности будем учитывать только две, наиболее часто используемые вычислительные операции, а именно: определение принадлежности объекта одному из дочерних октантов с условной стоимостью C_L и определение факта пересечения двух объектов со стоимостью C_I в том же самом предположении, что объекты геометрически представляются своими ограничивающими параллелепипедами. Затраты на конструирование новых октантов в структуре дерева, а также затраты на вставку и удаление ассоциированных объектов исключаются из рассмотрения. На самом деле эти операции всегда применяются в контексте основных вычислительных операций и поэтому неявно могут учитываться уже введенными константами.

3.1 Анализ стоимости развертывания регулярного октального дерева

Имеет место следующая лемма.

Лемма 3. Средняя стоимость развертывания регулярного октального дерева глубиной h для модельного набора данных $S(n,l)$ есть

$$Q_{deploy} = \frac{C_L n}{56(1-l)^3}(56h - 56 + 168l - 84 \cdot 2^h l - 56l^2 + 14 \cdot 2^{2h} l^2 + 8l^3 - 2^{3h} l^3)$$

Доказательство:

По лемме 1 математическое ожидание числа объектов в октантах i-го уровня и ниже ($1 \le i \le h$) есть

$$n_i = n\frac{(1-2^{i-1}l)^3}{(1-l)^3}$$

Именно столько объектов необходимо проанализировать для того, чтобы принять решение об их локализации на уровне i. Анализу не подлежат объекты, приписанные листовым октантам, поскольку необходимость их перераспределения в дочерних октантах отсутствует. Исходя из этих рассуждений, стоимость построения октального дерева определяется суммой затрат на локализацию объектов на уровнях $1,2,\dots,h-1$ и определяется следующей формулой:

$$Q_{deploy} = \sum_{i=1}^{h-1} C_L n\frac{(1-2^{i-1}l)^3}{(1-l)^3} = \frac{C_L n}{(1-l)^3}\sum_{i=1}^{h-1}(1-2^{i-1}l)^3$$

$$= \frac{C_L n}{56(1-l)^3}(56h - 56 + 168l - 84 \cdot 2^h l - 56l^2 + 14 \cdot 2^{2h} l^2 + 8l^3 - 2^{3h} l^3)$$

Лемма доказана.

Необходимо отметить, что в случае точечных данных ($l = 0$) стоимость построения октального дерева определяется выражением $Q_{deploy} = C_L n(h-1)$, что соответствует случаю полной локализации данных в листовых октантах и хорошо согласуется с известными результатами [5]. Заметим, что в силу леммы 1 количество объектов, подлежащих локализации, монотонно уменьшается с каждым уровнем и поэтому затраты на построение дерева всегда могут быть оценены как $Q_{deploy} \le C_L n(h-1)$ с учетом выражения для ожидаемой глубины дерева, основанного на результатах теоремы 1.

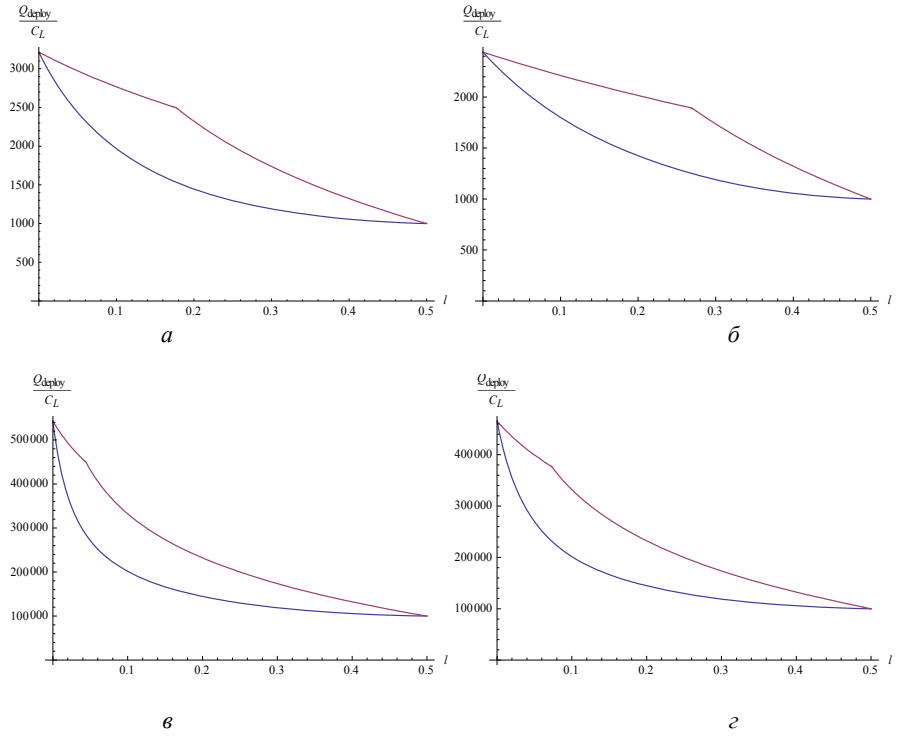


Рисунок 3. Зависимость стоимости развертывания дерева от габаритов объектов при разных соотношениях n и m. а)$n = 1000,\ m = 10$ б)$n = 1000,\ m = 50$ в)$n = 100000,\ m = 10$ г)$n = 100000,\ m = 50$

На рисунке 3 показаны кривые сложности развертывания октального дерева от габаритов объектов при разном соотношении общего числа объектов и порога наполненности октантов. Верхние кривые соответствуют приведенной приближенной оценке, нижние — воспроизводят функцию стоимости, полученную точно в предположениях леммы 3. Построенные графики показывают, что приближенная оценка отражает общий характер исследуемых зависимостей и отличается от функции стоимости на всем интервале изменения параметров не более, чем в два раза. Это дает основания применять ее на практике.

Теорема 2. Для модельного набора данных $S(n,l)$ регулярное октальное дерево $Octree(m)$ может быть построено в среднем за $Q_{deploy} = C_L q(n,m,l) \cdot n$, причем коэффициент $q(n,m,l)$ удовлетворяет следующим соотношениям:

$$q(n,m,l) = \left\lceil \frac{1}{3}\log_2 n - \frac{1}{3}\log_2 m \right\rceil \text{ при } l = 0,$$

$$q(n,m,l) \le \log_2 \frac{1}{l} \text{ при } l > 0$$

Доказательство:

Рассмотрим случай $l = 0$. По лемме 3

$$Q_{deploy} = \frac{C_L n}{56(1-l)^3}(56h - 56 + 168l - 84 \cdot 2^h l - 56l^2 + 14 \cdot 2^{2h} l^2 + 8l^3 - 2^{3h} l^3)$$

Подставляя $l = 0$, получим

$$Q_{deploy} = C_L n(h-1)$$

По теореме 1, при $l = 0$ выполняется следующее равенство:

$$h = \left\lceil \log_2 \frac{2}{l + (1-l)\sqrt[3]{\frac{m}{n}}} \right\rceil = \left\lceil \log_2 \frac{2}{\sqrt[3]{\frac{m}{n}}} \right\rceil = \left\lceil \log_2 \sqrt[3]{\frac{n}{m}} + 1 \right\rceil$$

$$= \left\lceil \frac{1}{3}\log_2 n - \frac{1}{3}\log_2 m \right\rceil + 1$$

Следовательно,

$$Q_{deploy} = C_L \left(\left\lceil \frac{1}{3}\log_2 n - \frac{1}{3}\log_2 m \right\rceil \right) \cdot n$$

Первая часть теоремы доказана.

Рассмотрим случай $l > 0$. Как было показано выше, имеет место общая оценка сложности развертывания дерева $Q_{deploy} \le C_L n(h-1)$. По теореме 1 глубина дерева всегда может быть ограничена как $h \le \log_2 \frac{2}{l}$. Следовательно, $Q_{deploy} \le C_L \log_2 \frac{1}{l} \cdot n$ и теорема доказана.

Теорема 2 имеет важные следствия, связанные с разным асимптотическим ростом сложности развертывания октального дерева для точечных и протяженных данных. Для первых ($l = 0$) сложность оценивается как $O(n\log_2 n)$, а для вторых ($l > 0$) — как $O(n)$. На рисунке 4 приведены графики сложности построения октального дерева от количества объектов в модельном наборе n при разных значениях порога наполненности октантов m, построенные на основе результатов леммы 3 для точечных (верхние кривые) и протяженных объектов (нижние кривые).

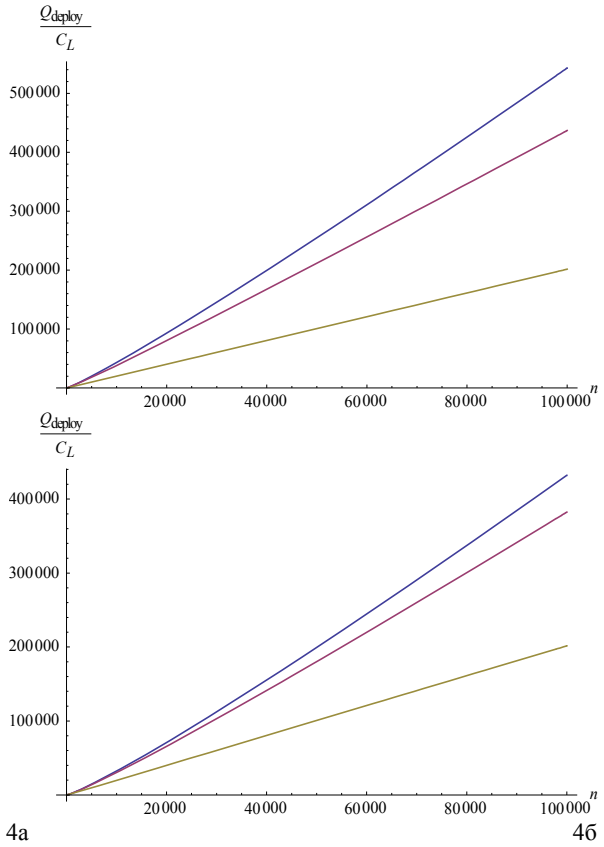


Рисунок 4. Зависимость стоимости построения октального дерева от количества объектов при различных габаритах объектов и параметра заполнения листовых октантов. а) $m = 10, l = 0{,}0.001{,}0.1$ б) $m = 100, l = 0{,}0.01{,}0.1$

Примечательно, что в исследуемом диапазоне размерности задачи (до ста тысяч объектов) все кривые выглядят схожим образом, несмотря на наличие логарифмического фактора в асимптотической оценке сложности для точечных объектов. Согласно приведенным графикам сложность построения дерева для протяженных объектов несколько ниже, чем для точечных данных. Это объясняется тем обстоятельством, что при одинаковом общем количестве протяженные объекты локализуются на более высоких уровнях октального дерева и не возникает необходимости в его дальнейшем развертывании.

3.2 Анализ сложности определения столкновений

Перейдем к оценкам затрат на выполнение типовых запросов, связанных с поиском столкновений объектов в пространственно-трехмерных сценах. Как

правило, точное пересечение двух геометрических объектов сложной формы требует большого объема вычислений. Поэтому более эффективной стратегией является предварительная локализация потенциальных столкновений с помощью простых тестов, основанных на сепарации пространства и пересечении ограничивающих объемов [6]. Точная процедура определения столкновений применяется лишь в случае положительного вердикта, что происходит в приложениях относительно редко и, как результат, существенно уменьшает общее время поиска столкновений. Эффект особенно ощутим для сцен, объекты которых геометрически представлены сложными аналитическими кривыми, поверхностями или полиэдрами с большим числом граней [7, 8].

Обсудим алгоритмические варианты использования октального дерева для локализации столкновений в сценах и получим оценки сложности в предположении, что сцена представлена модельным набором данных $S(n, l)$. Предположим, что октальное дерево развернуто и предстоит выявить все пары объектов, допускающие столкновения. В наивном алгоритме пришлось бы попарно пересечь все ограничивающие объемы объектов, что привело бы к затратам $\frac{1}{2} C_I n(n-1)$. Однако октальное дерево позволяет сделать это гораздо эффективнее с применением следующего алгоритма.

```
PROCEDURE CLASH (octant, result)
POINTER NODE octant
POINTER OBJECT PAIR COLLECTION result
FOR_EACH(OBJECT object1 IN OBJECTS(octant))
        {
        FOR_EACH(OBJECT object2 != object1 IN OBJECTS(octant))
                IF(IS_INTERSECTING(object1, object2))
                        INSERT_RESULT(result, PAIR(object1,
object2))

        FOR_EACH(NODE child IN CHILDREN(octant))
                CLASHCHILDREN(child, object1, result)
        }
FOR_EACH(NODE child IN CHILDREN(octant))
        CLASH(child, result)

PROCEDURE CLASHCHILDREN(octant, object, result)
POINTER NODE octant
POINTER OBJECT object
POINTER OBJECT PAIR COLLECTION result

IF(!IS_INTERSECTING(octant,object))
        RETURN
```



```
FOR_EACH(OBJECT o IN OBJECTS(octant))
        IF(IS_INTERSECTING(object, o))
                INSERT_RESULT(PAIR(object, o), result)

FOR_EACH(NODE child IN CHILDREN(octant))
        CLASHCHILDREN(child, object, result)
```

Алгоритм 2. Поиск столкновений при помощи регулярного октального дерева.

Процедура CLASH принимает в качестве параметра указатель на текущий октант, а также указатель на коллекцию результатов, каждый из которых представлен в виде пары объектов, имеющих потенциальную коллизию. В теле этой процедуры происходит поиск коллизий среди объектов, принадлежащих текущему октанту, а затем проверяются пересечения этих объектов с объектами, принадлежащими дочерним октантам при помощи вспомогательной процедуры CLASHCHILDREN, принимающей в качестве параметра объект и, посредствам поиска в глубину, ищущей все объекты, имеющие пересечения с данным. Затем процедура CLASH рекурсивно повторяется для всех дочерних октантов текущего октанта.

Перейдем к оценкам сложности для описанного алгоритма.

Лемма 4. Пусть регулярное октальное дерево $Octree(m)$ глубиной h развернуто для модельной сцены $S(n, l)$. Тогда столкновения объектов в ней могут быть локализованы за Q_{clash} в соответствии со следующей оценкой:

$$Q_{clash} \le \frac{C_I n^2}{2 \cdot 8^h (1-l)^6} (8 + l(357 \cdot 16^h l^3 - 28 \cdot 32^h l^4 + 64^h l^5 - 128(3 - 3l + l^2) + 56 \cdot 2^h \left(3 + 8l(3 - 3l + l^2)\right) - 42 \cdot 4^h l\left(23 + 16l(3 - 3l + l^2)\right) + 8^h l(114 + 1970l - 1483l^2 + 106l^3 + 105l^4)) + 14h(45 + 8l(3 - 3l + l^2))))))$$

Анализ совпадений точечных данных ($l = 0$) в сцене может быть проведен за $Q_{clash} \le \frac{C_I nm}{2}$.

Доказательство:

Описанный выше алгоритм 2 предполагает последовательный обход октантов всех уровней, начиная с самого верхнего. При обходе очередного октанта сначала анализируются возможные парные столкновения объектов внутри него. Для каждой ячейки это потребует $\frac{1}{2} C_I n_i(n_i - 1) \le \frac{C_I {n_i}^2}{2}$. Затем анализируются возможные столкновения внутренних объектов рассматриваемого октанта с объектами, приписанными дочерним октантам.

Заметим, что каждый внутренний объект пересекается не более, чем с восемью дочерними октантами на каждом следующем уровне октального дерева, поэтому затраты ограничиваются выражением $8C_I n_i \sum_{j=i+1}^{h} n_j$. Такая оценка априори не более чем в четыре раза превышает вычислительную сложность операции, поскольку в лучшем случае внутренний объект пересекается с двумя дочерними октантами. Таким образом, получаем следующую оценку стоимости локализации столкновений:

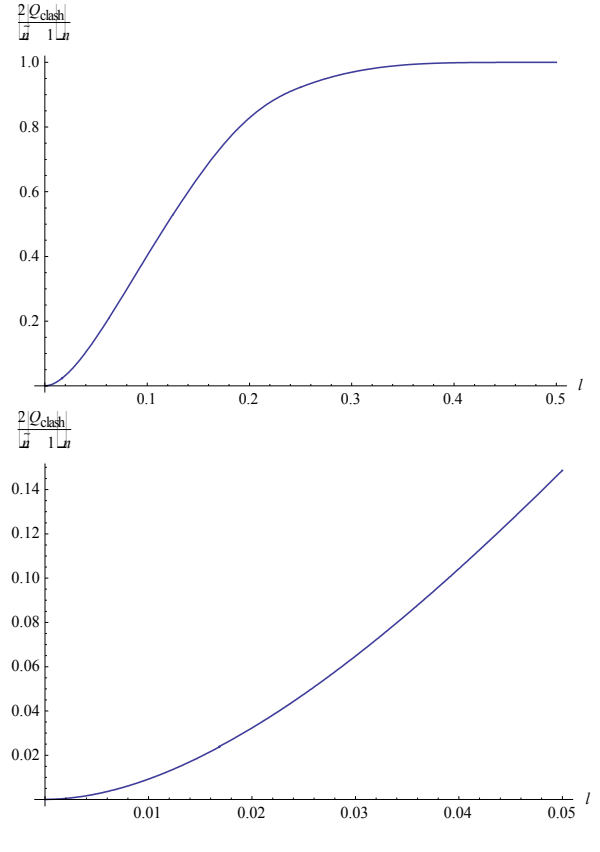
$$Q_{clash} \le \sum_{i=1}^{h} C_I 8^{i-1} n_i \left(\frac{n_i}{2} + 8 \sum_{j=i+1}^{h} n_j \right)$$

По лемме 2

$$n_i = \begin{cases} n \dfrac{\left(1-2^{i-1}l\right)^3 - (1-2^i l)^3}{8^{i-1}(1-l)^3}, i = 1..h-1 \\ n \dfrac{(1-2^{h-1}l)^3}{8^{h-1}(1-l)^3}, i = h \end{cases}$$

Подставляя n_i и выполнив суммирование рядов, получим требуемое выражение для общей оценки. Как и следовало ожидать, сложность локализации столкновений в общем случае пропорциональна квадрату числа объектов. Особый случай составляют точечные объекты ($l = 0$), для которых полученное выражение принимает вид $\frac{C_I n^2}{2 \cdot 8^{h-1}}$. По построению регулярного октального дерева для точечных объектов выполняется соотношение $\frac{n}{8^{h-1}} \le m$ и, следовательно, имеет место оценка $Q_{clash} \le \frac{C_I n m}{2}$. Что и требовалось доказать.

Важным следствием леммы является линейная сложность определения совпадений точечных данных с использованием регулярного октального дерева. Оценим эффект применения октального дерева для поиска столкновений протяженных объектов. На рисунке 5 представлены графики функции стоимости определения столкновений в модельном наборе данных в зависимости от относительных габаритов объектов. Графики построены на основе общей оценки леммы 4, а для наглядности значения стоимости отнесены к пессимистической оценке наивного поиска столкновений $\frac{1}{2} C_I n(n-1)$.



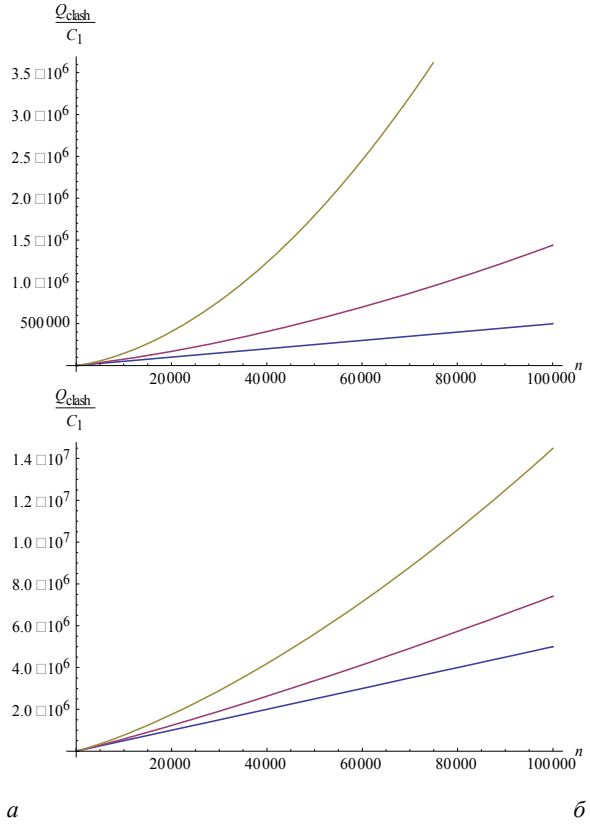
а *б*

Рисунок 5. Графики приведенной стоимости определения столкновений в модельном наборе данных в зависимости от габаритов объектов а) характер функции стоимости на всем диапазоне изменения габаритов объектов б) характер функции стоимости при относительно небольших габаритах объектов.

На графиках видно, что при больших габаритах объектов функция стоимости асимптотически стремится к пессимистической оценке. Для относительно небольших объектов стоимость поиска столкновений существенно ниже, однако быстро нарастает по мере увеличения габаритов объектов и пространственного заполнения сцены.

Проанализируем качественно, как стоимость поиска столкновений зависит от числа объектов и их размеров. На рисунках 6а и 6б приведены графики, иллюстрирующие характер зависимости при разных значениях порога наполнения октантов $m = 10$ и $m = 100$ и вариации габаритов объектов $l = 0$, $l = 0.001$ и $l = 0.003$. Видно, что сложность возрастает с увеличением числа объектов и их размеров. Нижние прямые на графиках соответствуют

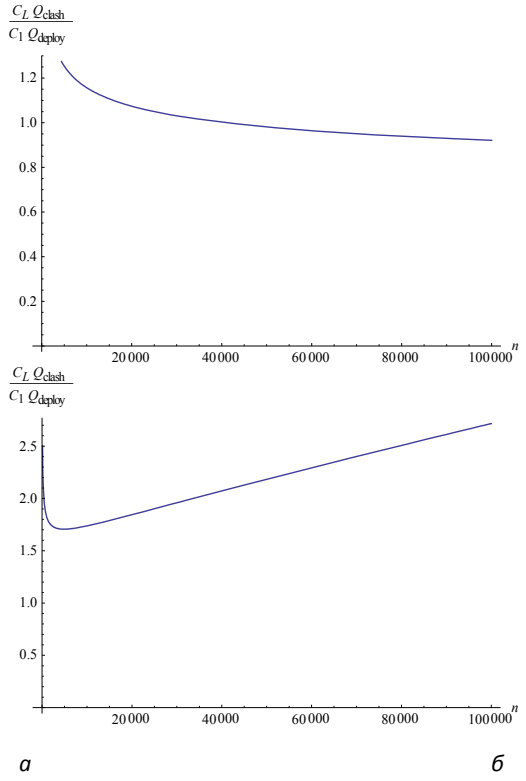
точечным данным, а средние и верхние кривые — протяженным объектам соответствующих размеров. Хотя анализ совпадений точечных данных имеет линейную сложность, а поиск столкновений протяженных объектов — квадратичную, для разреженных сцен различия оказываются не столь существенным даже для значительного числа объектов, составляющего на приведенных графиках 100000. Это обстоятельство оказывается чрезвычайно важным для применения обсуждаемых алгоритмов в индустриальных приложениях, оперирующими большими данными и подверженным быстрой деградации производительности даже при использовании алгоритмов невысокой полиномиальной сложности. Согласно приведенным графикам параметр наполнения октантов также влияет на стоимость поиска столкновений, однако принципиально не меняет характер исследуемых зависимостей.



а *б*

Рисунок 6. График стоимости локализации столкновений в зависимости от количества объектов при различных значениях габаритов ($l = 0,\ l = 0.001,\ l = 0.003$) а) при пороге наполнения октантов $m = 10$ б) при пороге наполнения октантов $m = 100$.

Таким образом, эффект использования регулярных октальных деревьев для быстрого поиска столкновений является значительным. Естественным является вопрос, в какой степени окупаются затраты на развертывание самих деревьев. Для этого построим семейство графиков для функции относительной сложности определения столкновений, приведенной к затратам на развертывание дерева. Построенные графики представлены на рисунке 7 и соответствуют значениям габаритов объектов $l = 0,\ l = 0.001,\ l = 0.01$ и $l = 0.03$ при значении порога наполнения октантов $m = 10$. На графике 7а, соответствующем точечным данным, видно, что затраты на развертывание октального дерева приблизительно соотносятся с затратами на анализ совпадений, а с учетом достигнутого эффекта в производительности всегда окупятся. Различия еще более ощутимы для сцен с протяженными объектами. Затраты на построение дерева становятся пренебрежительно малы по отношению к затратам на определение столкновений. Однако следует иметь в виду, что и эффект от использования октальных деревьев существенно падает с заполнением сцены и необходимостью применения наивного алгоритма для пересечения всех пар объектов в октантах верхних уровней.



а *б*

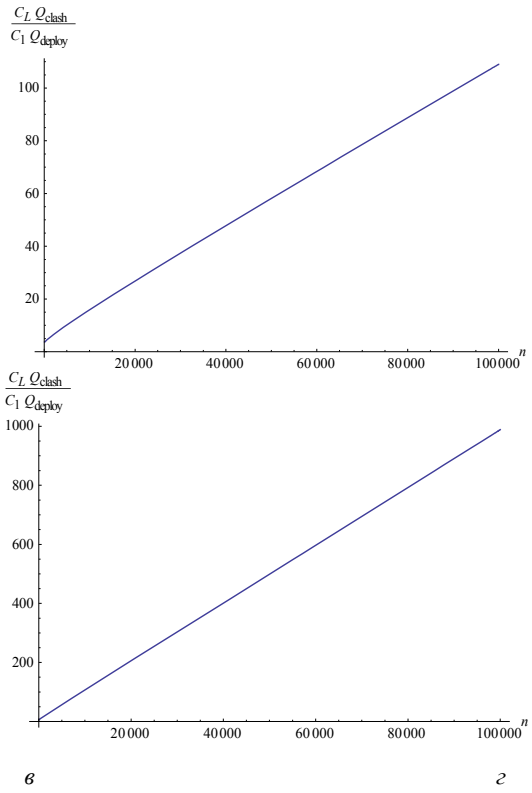
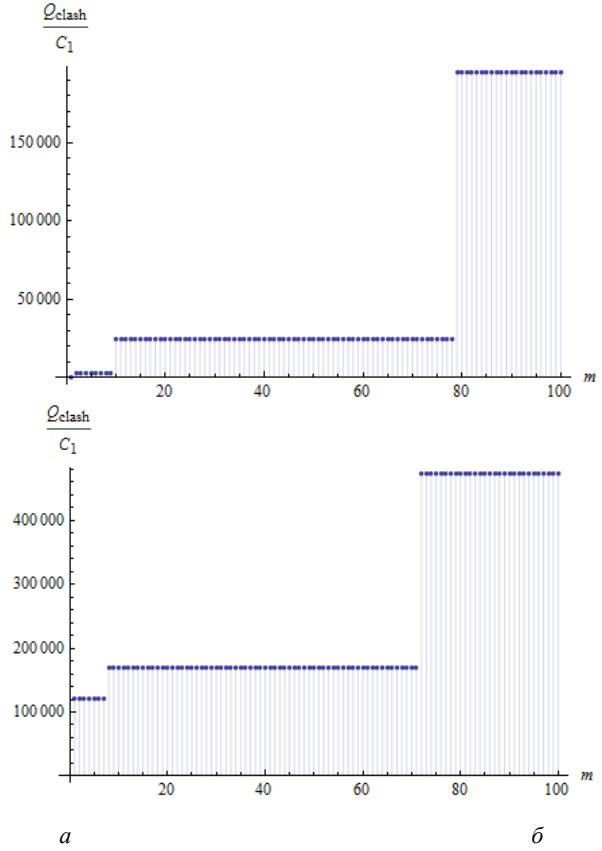
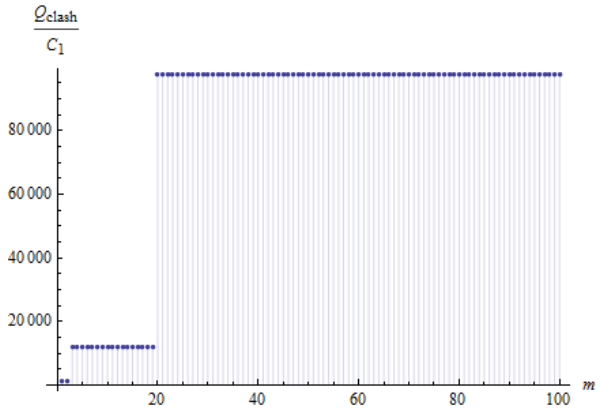


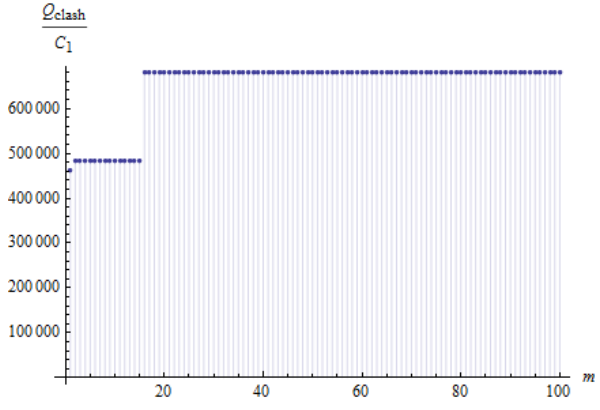
Рисунок 7. График стоимости определения столкновений относительно затрат на развертывание регулярно октального дерева при различных габаритах объектов а)$l = 0$ б)$l = 0.001$ в)$l = 0.01$ г)$l = 0.03$

Важным представляется вопрос о выборе порогового значения наполнения октантов m, обеспечивающего минимальные затраты на определение столкновений. Уменьшение порога приводит к неизбежному перераспределению объектов в листовых октантах следующего уровня дерева. С одной стороны, это приводит к более высокой фрагментации объектов и уменьшению затрат на определение столкновений внутри отдельных листовых октантов. С другой стороны, с увеличением глубины дерева возрастают расходы на анализ столкновений объектов, локализованных на верхних уровнях дерева, с объектами, перераспределенными в листьях. Однако проведенный анализ показывает, что второй фактор не является доминирующим и выбор минимального порога всегда приводит к уменьшению затрат на определение столкновений. На рисунке 8 представлено семейство графиков для функции затрат в зависимости от порога наполнения m. Графики построены для разных сочетаний числа объектов и их

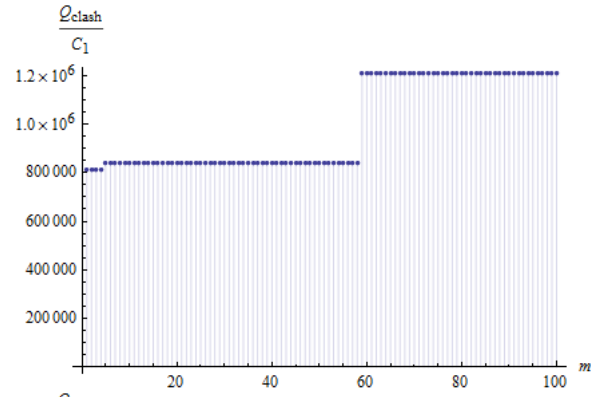
относительных габаритов. Как следует из анализа графиков, выбор $m = 10$ является вполне оправданным для всех проанализированных вариантов. Данное значение всегда обеспечивает низкие затраты на определение столкновений. В тоже время, он исключает излишне детальную декомпозицию пространства и неизбежные расходы по памяти на хранение более глубокого представления октального дерева. Действительно, с каждым новым уровнем дерева количество октантов увеличивается в восемь раз, что приводит к быстрому исчерпанию объема доступной памяти приложениями.

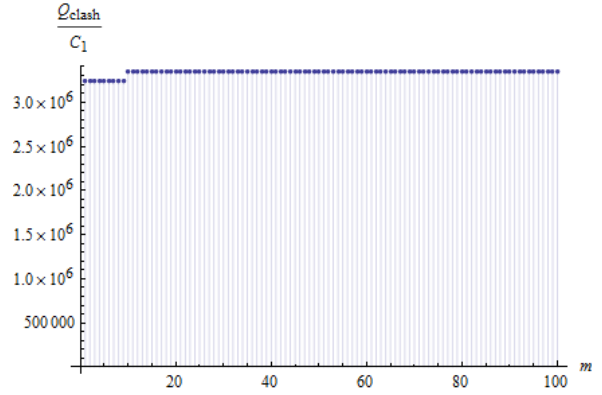






в *г*





д *е*

Рисунок 8. График функции затрат на определение столкновений в зависимости от порога наполнения октантов при различных параметрах модельной сцены а) $n = 5000, l = 0$ *б)* $n = 5000, l = 0.01$ *в)* $n = 10000, l = 0$ *г)* $n = 10000, l = 0.01$ *д)* $n = 5000, l = 0.03$ *е)* $n = 10000, l = 0.03$

3.3 Анализ сложности выборки объектов по заданной области

Перейдем к оценкам затрат на выборку объектов, принадлежащих заданной пространственной области или имеющих непустое пересечение с ней. Для определенности будем считать, что область представляет собой куб с относительным размером ребер L, приведенным к габаритам ограничивающего параллелепипеда всего набора данных. Ниже приведен псевдокод алгоритма выборки.

PROCEDURE CLIP (octant, box, result)
POINTER NODE octant
BOX box

```
POINTER OBJECT COLLECTION R

IF(!IS_INTERSECTING(octant,box))
        RETURN

FOR_EACH(OBJECT o IN OBJECTS(octant))
        {
        IF(IS_INTERSECTING(o, box))
                ADD (o, result)
        }

FOR_EACH(NODE child IN CHILDREN(octant))
        CLIP(child, box, result)
```

Алгоритм 3. Выборка объектов по заданной области на основе регулярного октального дерева.

В теле процедуры CLIP октальное дерево обходится с самого верхнего уровня и октанты подвергаются рекурсивной проверке на пересечение с областью выборки. В случае положительного вердикта аналогичной проверке подвергаются все объекты, приписанные к текущему октанту, а также все дочерние октанты, в случае отрицательного вердикта — поиск в глубину текущего октанта прекращается. Объекты, приписанные октантам с ненулевым пересечением области выборки, группируются и формируют окончательный результат.

Определим характерный уровень локализации области выборки как $H = \left\lceil \log_2 \frac{1}{L} \right\rceil$.

Лемма 5. Пусть регулярное октальное дерево глубиной h развернуто для модельной сцены $S(n, l)$. Тогда выборка объектов по области с уровнем локализации H может быть проведена в среднем за Q_{clip} в соответствии со следующей оценкой:

$$Q_{clip} \leq \begin{cases} C_I n \dfrac{64 - 224 \cdot 2^h l + 336 \cdot 4^h l^2 + 8^h (11l - 81l^2 + (113 - 56h) l^3)}{8^h (1-l)^3} \\ +8C_I(h-1), h \leq H \\ C_I n \dfrac{64 - 224 \cdot 2^H l + 336 \cdot 4^H l^2 + 8^H (11l - 81l^2 + (113 - 56H) l^3)}{8^H (1-l)^3} \\ +C_I \dfrac{8^{h-H+2} + 56H - 120}{7}, h > H \end{cases}$$

Доказательство:

В соответствии с приведенным алгоритмом вначале определяются октанты, принадлежащие области выборки, а затем выбираются объекты, имеющие ненулевое пересечение с ней. Заметим, что на первом уровне необходимо проанализировать лишь один октант и все приписанные ему объекты. На каждом следующем уровне $i = 2 \dots H$ вплоть до уровня локализации области анализу подлежит не менее одного и не более восьми октантов. На оставшихся уровнях $i = H+1 \dots h$ достаточно проанализировать не более 8^{i-H+1} октантов. С учетом математического ожидания числа объектов n_i на каждом i-уровне дерева вычислительная сложность выборки может быть оценена как

$$Q_{clip} \leq \begin{cases} C_I \left(n_1 + 8 \sum_{i=2}^{H} (n_i + 1) + \sum_{i=H+1}^{h} 8^{i-H+1} (n_i + 1) \right), h > H \\ C_I \left(n_1 + 8 \sum_{i=2}^{h} (n_i + 1) \right), h \leq H \end{cases}$$

Отметим, что такая оценка не более чем в 8 раз превосходит минимальные затраты на выполнение выборки. Используя результаты леммы 2 для математического ожидания числа объектов n_i и проводя суммирование рядов, имеем при $h \leq H$

$$Q_{clip} \leq C_I n \frac{64 - 224 \cdot 2^h l + 336 \cdot 4^h l^2 + 8^h (11l - 81l^2 + (113 - 56h) l^3)}{8^h (1-l)^3} + 8C_I(h-1)$$

В противоположном случае $h > H$ имеем оценку

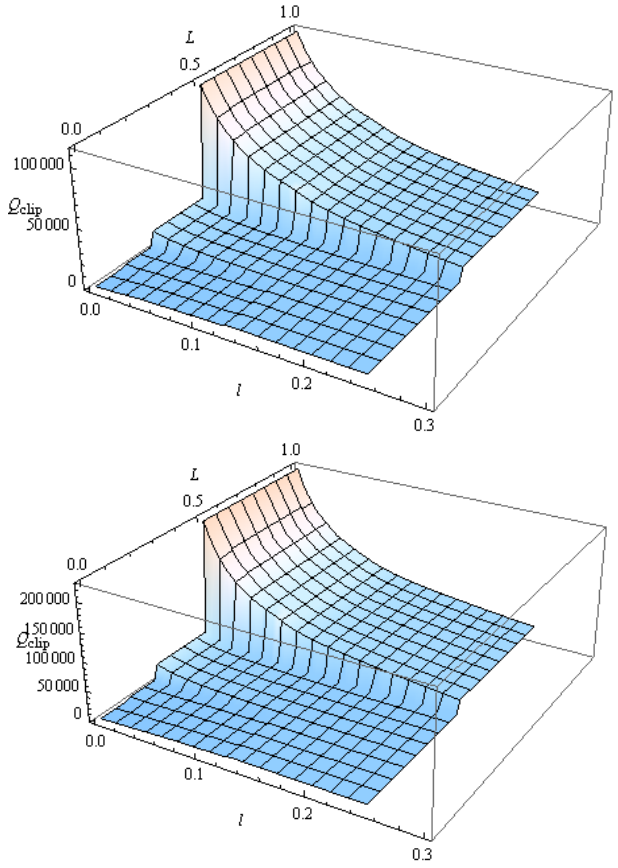
$$Q_{clip} \leq C_I n \frac{64 - 224 \cdot 2^H l + 336 \cdot 4^H l^2 + 8^H (11l - 81l^2 + (113 - 56H) l^3)}{8^H (1-l)^3} + C_I \frac{8^{h-H+2} + 56H - 120}{7}$$

Что и требовалось доказать.

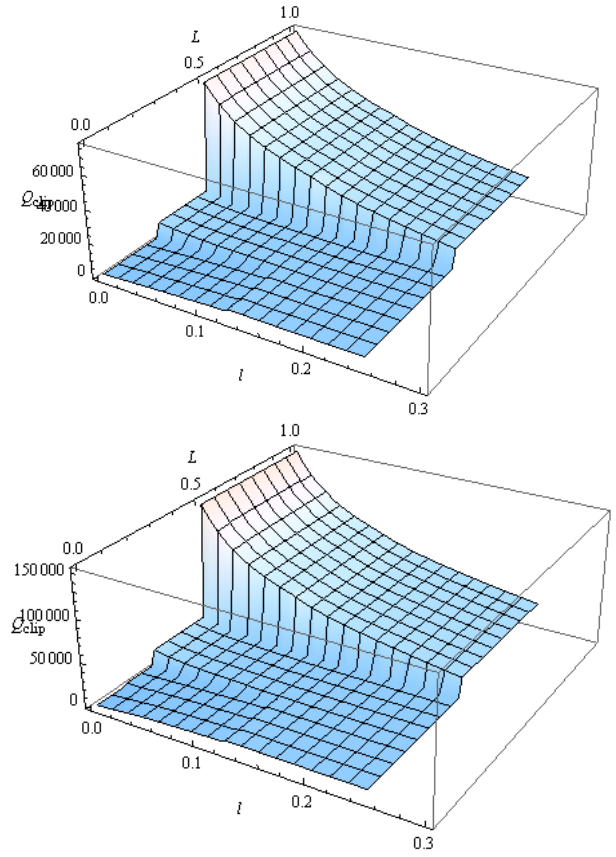
Заметим, что при $H = h$ полученные выражения полностью совпадают. Первые слагаемые в обеих оценках выражают затраты на поиск объектов, принадлежащих области выборки или пересекающие ее, вторые — затраты на

предварительную локализацию октантов, в которых такие объекты могут быть найдены. Полученные результаты согласуются с ранее полученными оценками для точечных данных [9].

На рисунке 9 представлены семейства графиков, иллюстрирующие зависимость стоимости выборки объектов от габаритов объектов и размера области выборки. Как видно, стоимость выборки растет с увеличением размера области выборки и при $L > \frac{1}{4}$ использование октального дерева теряет всякий смысл. При меньших размерах применение октального дерева вполне оправданно, однако рост габаритов объектов может нивелировать суммарный эффект.



а *б*



в *д*

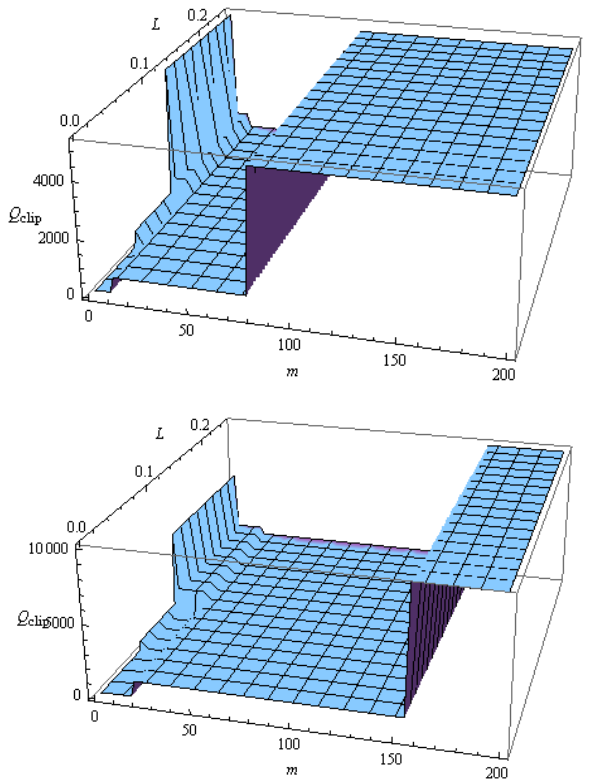
Рисунок 9. Графики стоимости выборки от габаритов объектов и размера области выборки а) $n = 5000, m = 5$ *б)* $n = 10000, m = 5$ *в)* $n = 5000, m = 10$ *г)* $n = 10000, m = 10$

Обсудим вопрос о разумном выборе порога наполнения октантов. Для этого построим соответствующие графики зависимости стоимости выборки от порога наполнения и размеров области выборки. Семейства графиков на рисунке 10 воспроизводят эти зависимости при разных параметрах модельной сцены, а именно: при вариации числа объектов и их размеров. Как видно из этих графиков, затраты на выполнение выборки возрастают с ростом габаритов области выборки. Вместе с тем, зависимость от порога наполнения октантов носит более сложный характер. Так, при больших габаритах области выборки ($L \geq \frac{1}{16}$) выгодны большие значения порога заполнения, поскольку в

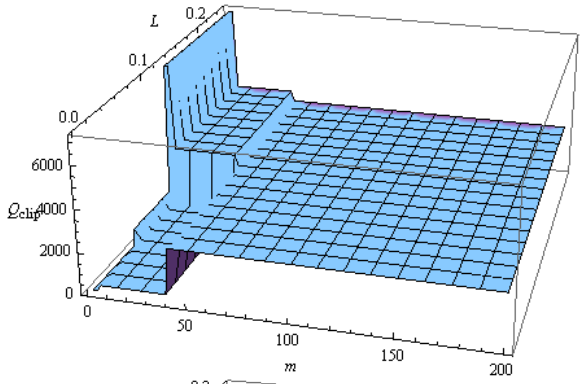
этом случае в стоимости выборки превалирует компонент, связанный с предварительной локализацией октантов, в которых могут содержаться интересующие нас объекты. С другой стороны, при малых габаритах области выборки ($L < \frac{1}{16}$), наибольшее влияние оказывает слагаемое соответствующее затратам на поиск объектов, которые, в свою очередь, пропорциональны значению m.

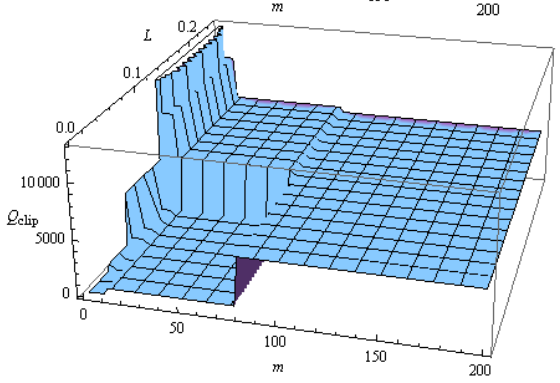
В исследуемом диапазоне изменения параметров сцены выбор $m = 50$ является вполне приемлемым, поскольку минимизирует затраты на исполнение типовых запросов выборки и при этом не приводит к избыточному расходованию памяти на развертывание глубоких октальных деревьев.

Тем не менее, выбор порога наполнения в общем случае является нетривиальным. Во-первых, рассмотренный модельный набор данных лишь эмулирует упрощенную постановку задачи, связанной с моделированием пространственно-трехмерных сцен. В реальных приложениях сцены могут содержать объекты, которые имеют существенно различные габариты и неравномерно распределены по пространству. Полученные строгие оценки применимы лишь к модельному набору данных, однако правомерность их использования для произвольных сцен, видимо, должна подкрепляться вычислительными экспериментами. Тем не менее, по мере приближения к модельному случаю, мы можем рассчитывать и на адекватность полученных теоретических результатов. Во-вторых, неопределенность с выбором порога наполнения усугубляется тем обстоятельством, что приложения могут консолидировать многие функции одновременно, а их эффективная реализация может предполагать использование совершенно разных значений порога, как в рассмотренных задачах локализации столкновений и выборки по области. В этих случаях, видимо, следует руководствоваться частотой применения отдельных функций и выбора порога, обеспечивающего их равномерно высокую эффективность.

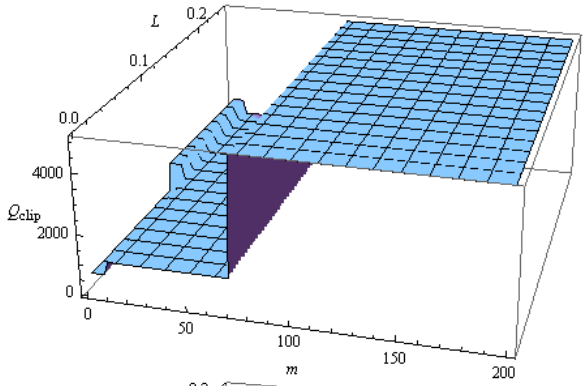


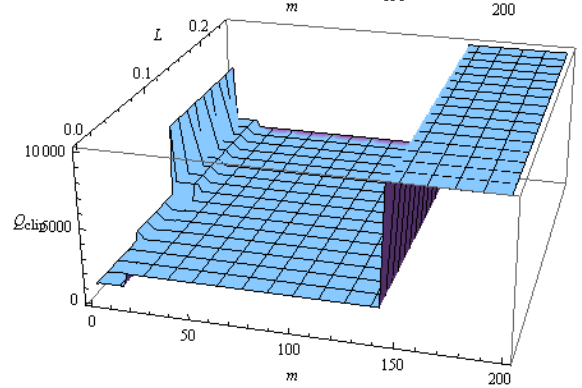
а *б*


L
0.2
0.1
0.0
6000
4000
2000
0
Q_{clip}
0
50
100
150
200
m

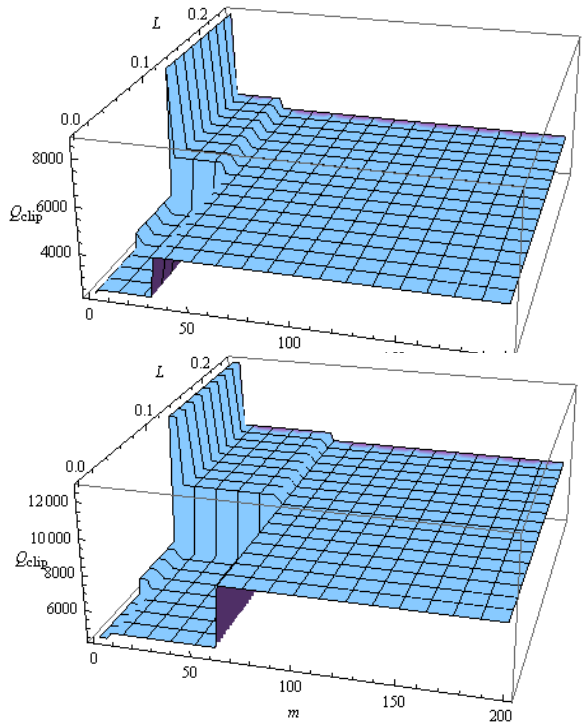

L
0.2
0.1
0.0
10 000
5000
0
Q_{clip}
0
50
100
150
200
m

в *г*


L
0.2
0.1
0.0
4000
2000
0
Q_{clip}
0
50
100
150
200
m


L
0.2
0.1
0.0
10 000
5000
0
Q_{clip}
0
50
100
150
200
m

д *е*



ж *з*

Рисунок 10. Графики зависимости стоимости выборки от размеров области и порога наполнения октантов при разных параметрах модельной сцены. а) $n = 5000, l = 0$ *б)* $n = 10000, l = 0$ *в)* $n = 20000, l = 0$ *г)* $n = 40000, l = 0$ *д)* $n = 5000, l = 0.01$ *е)* $n = 10000, l = 0.01$ *ж)* $n = 20000, l = 0.01$ *з)* $n = 40000, l = 0.01$

4. Заключение

Таким образом, рассмотрено важное семейство методов декомпозиции, основанное на регулярных октальных деревьях и обеспечивающее эффективную индексацию, быстрый поиск и анализ больших пространственных данных. Для алгоритмов определения столкновений и выборки объектов по заданной области получены оценки сложности в среднем, которые улучшают известные пессимистические результаты и служат теоретическим обоснованием их применения к более широкому классу приложений. В частности, обсуждены возможности применения рассмотренных алгоритмов к приложениям моделирования сложных пространственно-трехмерных сцен с объектами, имеющими протяженные границы. Существенной стороной проведенного исследования является

попытка определить границы применимости методов декомпозиции, а также выработать практические рекомендации по использованию конкретных алгоритмов в широком диапазоне изменения характеристик пространственных данных.

Список литературы

[1]. M. Cai., P. Revesz, Parametric rectangles: an index structure for moving objects, в *Proceedings of the 10th COMAD International conference on management of data*, 2000.

[2]. V. Semenov, K. Kazakov, S. Morozov, O. Tarlapan, V. Zolotov и T. Dengenis, 4D modeling of large industrial projects using spatio-temporal decomposition, в *eWork and eBusiness in Architecture, Engineering and Construction*, London, UK, pp. 89-95, 2010.

[3]. В. Золотов и В. Семенов, Современные методы поиска и индексации многомерных данных в приложениях моделирования больших динамических сцен, *Труды Института системного программирования,* (24), pp. 381-416, 2013. DOI: 10.15514/ISPRAS-2013-24-17.

[4]. G. Kedem, The quad-CIF tree: a data structure for hierarchical on-line algorithms, в *Proceedings of the 19th Design Automation Conference*, pp. 352-357, 1992.

[5]. B. J. Finkel R.A., Quad trees: a data structure for retrieval on composite keys, *Acta Informatica,* 1(4), pp. 1-9, 1974.

[6]. V. Semenov, K. Kazakov, V. Zolotov, H. Jones и S. Jones, Combined strategy for efficient collision detection in 4D planning applications, в *Computing in Civil and Building Engineering*, Nottingham, UK, pp. 31-39, 2010.

[7]. S. Gottschalk, M. C. Lin и D. Manocha, OBB Tree: a hierarchical structure for rapid interference detection, в *Proceedings of the SIGGRAPH'96 Conference*, New Orleans, USA, pp. 171-180, 1996.

[8]. S. Gottschalk, Collision queries using oriented bounding boxes, Chapel Hill: The University of North Carolina, 2000.

[9]. H. Samet, Foundations of Multidimentional and Metric Data Structures, San Francisco: Morgan Kaufmann, 2006.

On application of spatial decomposition method for large data sets indexing.

Zolotov V.A,. Semenov V.A. (ISP RAS, Moscow, Russia)

Annotation. The paper is dedicated to theoretical research of spatial indexing methods in conformity to three dimensional scenes arising in CAD/CAM systems, robotics, virtual and augmented reality applications. Special attention is given to the decomposition methods based on regular dynamic octrees. A particular version of the octrees is described and investigated to satisfy to the efficiency requirements for typical spatial queries in complex dynamic scenes with the extended borders objects. To perform the needed complexity analysis, the developed octree structure is analyzed against typical spatial queries like collision detection, frustum culling. To obtain more relevant estimates of the complexity on average rather than known pessimistic estimates in worst case, model scene datasets have been introduced. They assume uniform distribution of equal size objects over the scene volume and, being parametric, allow simulation of wide range of industry meaningful scenes. Some auxiliary lemmas about octree depth and distribution of the scene objects over octree levels have been proven based on probabilistic analysis. Final theorem statements provides for the complexity estimates of query evaluation for the model scenes. The obtained results show that asymptotic complexity grows with the object sizes and the scene occupancy. However, the estimates on average improve known results in worst case and can be considered as a theoretical background to introduce the investigated indexing structures to practical applications.

Keywords: spatial indexing, octrees, computational complexity

References

[1]. Cai M., Revesz P. Parametric rectangles: an index structure for moving objects. 2000. Proceedings of the 10th COMAD International conference on management of data.

[2]. Semenov V.A., Kazakov K.A., Morozov S.V., Tarlapan O.A., Zolotov V.A., Dengenis T. 4D modeling of large industrial projects using spatio-temporal decomposition. [ed.] K. Menzel и R. Scherer. London, UK : CRC Press, Taylor & Francis Group, 2010. eWork and eBusiness in Architecture, Engineering and Construction. pp. 89-95.

[3]. Zolotov V.A., Semenov V.A. Sovremennye metody poiska i indeksatsii mnogomernykh dannykh v prilozheniyakh modelirovaniya bol'shikh dinamicheskikh stsen [Advanced indexing methods for large multidimentional data in complex dynamic scenes]. Trudy ISP RAN [The Proceedings of ISP RAS], 2013, vol. 24, pp. 381-416. DOI: 10.15514/ISPRAS-2013-24-17. (in Russian)

[4]. Kedem G. The quad-CIF tree: a data structure for hierarchical on-line algorithms. 1992. Proceedings of the 19th Design Automation Conference. pp. 352-357.

[5]. Finkel R.A., Bentley J.L. Quad trees: a data structure for retrieval on composite keys. Acta Informatica. 1974, vol. 4, 1, pp. 1-9.

[6]. Semenov V.A., Kazakov K.A., Zolotov V.A., Jones H., Jones S. Combined strategy for efficient collision detection in 4D planning applications. [ed.] W. Tizani. Nottingham, UK : Nottingham University Press, 2010. Computing in Civil and Building Engineering. pp. 31-39. ISBN 978-1-907284-60-1.

[7]. Gottschalk S., Lin M. C., Manocha D. OBB Tree: a hierarchical structure for rapid interference detection. New Orleans, USA, 1996. Proceedings of the SIGGRAPH'96 Conference. pp. 171-180.

[8]. Gottschalk S. Collision queries using oriented bounding boxes. Chapel Hill : The University of North Carolina, 2000.

[9]. Samet H. Foundations of Multidimentional and Metric Data Structures. San Francisco : Morgan Kaufmann, 2006.