*На правах рукописи*

**Беляевский Кирилл Олегович**

**МЕТОДЫ И АЛГОРИТМЫ ФОРМИРОВАНИЯ ОКТОДЕРЕВА**

**БОЛЬШОГО ОБЛАКА ТОЧЕК ЛАЗЕРНОГО СКАНИРОВАНИЯ**

Специальность 05.13.01 – Системный анализ, управление и обработка

информации (технические системы)

**АВТОРЕФЕРАТ**

диссертации на соискание ученой степени

кандидата технических наук

Санкт-Петербург – 2019

Работа выполнена в федеральном государственном автономном образовательном учреждении высшего образования «Санкт-Петербургский политехнический университет Петра Великого».

|  |  |
| --- | --- |
| Научный  руководитель: | доктор технических наук, профессор  **Мелехин Виктор Федорович** |
| Официальные  оппоненты: |  |
| Ведущая  организация: | Акционерное общество «Концерн «НПО «Аврора»,  г. Санкт-Петербург |

Защита диссертации состоится «\_\_\_»\_\_\_\_\_\_\_2019 г. в \_\_\_ часов \_\_\_ минут на заседании диссертационного совета Д.002.199.01, созданного на базе Федерального государственного бюджетного учреждения науки Санкт-Петербургского института информатики и автоматизации Российской академии наук по адресу: 199178, Санкт-Петербург, 14 линия В.О., 39.

С диссертацией можно ознакомиться в библиотеке и на сайте Федерального государственного бюджетного учреждения науки Санкт-Петербургский институт информатики и автоматизации Российской академии наук http://www.spiiras.nw.ru.

Автореферат разослан «\_\_\_»\_\_\_\_\_\_\_\_\_\_2019г.

|  |  |
| --- | --- |
| Ученый секретарь  диссертационного совета Д 002.199.01  кандидат технических наук | А.А. Зайцева |

Общая характеристика работы

Актуальность темы диссертации. В настоящее время активно развивается область измерительных технологий. Результатом такого развития является появление устройств, позволяющих собирать пространственную информацию об объектах местности, рельефе, сооружениях на качественно новом уровне. Такая информация в настоящее время широко используется для решения инженерных, геодезических и прочих задач.

Одним из наиболее прогрессивных методов сбора цифровой пространственной информации на текущее время является лазерное сканирование (ЛС). Главными достоинствами устройств лазерного сканирования является высокая точность, возможность автоматизации процесса сбора данных, хорошая плотность измерений. Современные лазерные сканеры могут быть компактными и мобильными, их точность и скорость работы на текущий момент выросла на порядок, а в некоторых случаях на несколько порядков.

Принцип работы технологии лазерного сканирования основан на испускании лазерного импульса, и измерении расстояния от источника до объекта сканирования за счет регистрации временной задержки или сдвига фаз до приема отраженного от объекта импульса. Имея информацию о местоположении сканера и направлении импульса можно определить координаты точки объекта, отразившего импульс. Результатом таких измерений является трехмерное облако точек, с высокой точностью моделирующее объект исследования.

Современные лазерные сканирующие системы позволяют производить миллионы измерений в секунду, а размеры получаемых облаков точек могут достигать нескольких сотен гигабайт, что предъявляет высокие требования к вычислительным ресурсам при обработке таких данных. В результате имеет место очевидное противоречие между быстрым развитием технологий ЛС и возможностями обработки результатов, на разрешение которого направлено настоящая работа.

Используемые стандартные методы обработки данных лазерного сканирования, а также высокие требования к вычислительным ресурсам, не позволяют в полной мере использовать возможности ЛС. В настоящее время обработка результатов лазерного сканирования зачастую связана с разработчиком/производителем оборудования для ЛС и ПО для обработки результатов. Среди открытого ПО наблюдается нехватка инструментов для обработки облаков точек, чей размер превышает доступную оперативную память. Так как объемы доступной оперативной памяти могут варьироваться, и зачастую не способны вместить требуемое количество данных, а рост объемов оперативной памяти значительно отстает от роста объемов облаков точек, вызванным развитием технологий ЛС, задача ограничения потребляемой оперативной памяти при обработке облаков точек является актуальной.

В диссертационной работе представлены результаты разработки методов и алгоритмов решения задачи обработки больших облаков точек при помощи октодерева и различных подходов к выделению (аллокации) памяти, что позволяет разрешить противоречие между все возрастающим объемом данных ЛС и ограничений по оперативной памяти.

**Степень разработанности темы исследования.** Октодерево широко используется в различных областях, связанных с большими объемами трехмерных данных, начиная с момента своего представления Мигером Д. в начале 80-х годов. В середине 80-х были рассмотрены точки в качестве примитивов визуализации для моделей с замкнутыми поверхностями в работе Левой М. и Уитед Т. Ими же было установлено, что почти любой вид геометрического представления может быть преобразован в точечную модель. Русинкевич Ш. и Левой М. в 2000 г. используют сетки в качестве источников для облаков точек, используя позиции вершин сетки в качестве точек выборки. Для выполнения пространственного поиска они используют иерархию вложенных сфер.

Технология лазерного сканирования получила широкое распространение начиная с 2000 г. На текущий момент, получение сцен реального мира с использованием технологии трехмерного сканирования является стандартной техникой, область применения которой варьируется от сканирования механических деталей, до сканирования памятников исторического наследия и городов. При таком применении создаются огромные объемы трехмерных данных, представленных неструктурированными облаками точек.

В 2002 г. Цвикер М., Паули М., Нолл О. и Гросс М. предложили подход для интерактивного редактирования поверхности, представленной небольшим облаком точек, целиком помещающимся в оперативную память. Однако редактирование и обработка геометрии становятся намного более проблематичными, если наборы данных выходят за пределы основной памяти. Из свойств оборудования компьютеров поколения видно, что существует огромный разрыв в производительности между доступом к данным на внешних устройствах памяти и доступом к данным в основной памяти. Наибольший разрыв в производительности приходится на время доступа, поэтому количество обращений к внешней памяти должно быть как можно меньше.

В 2008 г. Виттер Д. С. предполагает, что обработка наборов данных на внешних системах хранения может быть выполнена эффективно, если алгоритм загружает порции данных в память перед их обработкой и работает только с этими порциями. Самет Х. упоминает, что хранение недавно использованных фрагментов при помощи системы кеширования может еще больше уменьшить количество обращений к диску при обработке данных. Структуры данных, поддерживающие эти методы, должны группировать точки данных в сегментах, поскольку эти сегменты можно легко использовать в качестве фрагментов для записи/чтения данных которые передаются на внешнюю систему хранения, что снижает количество обращений к диску по сравнению с переносом отдельных точек.

В 2014 г. Шибауер К. предлагает подход к хранению данных облака точек в октодереве с возможностью интерактивного редактирования облака точек, в которой обработка больших объемов данных осуществлялась за счет использования системы кеширования точек на вторичную систему хранения. На основе его работ, Шуц М. в 2016 г. предлагает систему визуализации больших облаков точек для веб-браузеров.

Вместе с тем, к сожалению, на текущий момент существующие системы обработки больших облаков точек имеют узкую специализацию, а в процессе формирования страдают от недостатка производительности и создают избыточное количество файлов, перегружающих файловую систему. Также отсутствует общая поддержка обработки больших облаков точек в существующих широко используемых открытых программах и библиотеках. Таким образом, задача формирования октодерева для обработки больших облаков точек, позволяющего сохранить приемлемую производительность обработки, сократить количество создаваемых файлов, а также обеспечить простой доступ к данным и возможность интеграции с другими программными решениями, является перспективным направлением.

Целью работы является разработка методов и алгоритмов структурирования облака точек лазерного сканирования и организация хранения и доступа к этой информации при превышении ею доступного объема оперативной памяти. Для достижения поставленной цели в диссертационной работе поставлены и решены следующие задачи:

1. Анализ методов и алгоритмов обработки больших облаков точек, существующих структур представления информации из облака точек (структур разбиения пространства) и обоснование выбора типа структуры.
2. Анализ модели задач обработки, получение оценок сложности доступа к информации из облака точек, определение требований к предварительному структурированию информации с целью уменьшения сложности, определение критериев эффективности формируемых структур данных.
3. Разработка структуры данных и алгоритмов формирования октодерева для обработки больших облаков точек, а также системы управления процессами загрузки/выгрузки данных на устройство долговременного хранения при превышении обрабатываемой информацией объема основной памяти.
4. Разработка программного обеспечения для обработки (структурирования информации) облаков точек ЛС с применением предложенных алгоритмов и обработки облаков точек ЛС с применением сторонней библиотеки в интеграции с предложенными алгоритмами.
5. Разработка системы показателей, характеризующих процесс структурирования данных, превышающих объем основной памяти, и разработка вычислительных экспериментов для получения этих показателей, а также методики выбора параметров процесса формирования октодерева, необходимых для разработки программного обеспечения.

Объектом исследования является процесс обработки больших облаков точек лазерного сканирования.

Предметом исследования являются методы и алгоритмы, позволяющие выполнять обработку больших облаков точек при ограничении потребляемой оперативной памяти, за счет вторичного устройства хранения.

Основные методы исследования. Для решения поставленных задач в работе используется системный подход, теория множеств и отношений. При разработке архитектуры программного обеспечения был применен объектно-ориентированный подход.

Положения, выносимые на защиту. На основе проведенной работы и ее экспериментальной апробации на защиту выносятся следующие положения:

1. Метод формирования октодерева на базе двухуровневой асинхронной системы кеширования, позволяющий выполнять обработку облаков точек в условиях ограничений по оперативной памяти, обеспечивающий снижение количества создаваемых файлов в процессе построения.
2. Метод формирования октодерева с использованием механизма отображения памяти, позволяющий выполнять обработку облаков точек в условиях ограничений по оперативной памяти, обеспечивающий прямой доступ к данным и возможность их интерактивной модификации, а также сокращающий количество создаваемых файлов в процессе формирования и использования октодерева.
3. Алгоритм динамической аллокации на отображаемой памяти, позволяющий уменьшить количество создаваемых файлов в процессе обработки большого облака точек до одного.
4. Способ обеспечения возможности обработки больших облаков точек в сторонних библиотеках для linux систем путем интеграции системы аллокации на отображаемой памяти.

Научная новизна предлагаемой диссертации состоит в следующем:

1. Предложена иерархическая модель октодерева с использованием целочисленной арифметики.
2. Предложен метод формирования октодерева на базе двухуровневой системы кеширования с асинхронной выгрузкой, позволяющего выполнять обработку облаков точек при ограничениях оперативной памяти.
3. Предложен метод формирования октодерева с использованием механизма отображения памяти, позволяющего выполнять обработку облаков точек при ограничениях оперативной памяти.
4. Предложен алгоритм динамической аллокации на отображаемой памяти.
5. Предложен способ интеграции системы аллокации на отображаемой памяти в сторонние библиотеки для linux систем.

**Теоретическая значимость работы.** Разработана иерархическая модель октодерева на основе целочисленной арифметики, позволяющая повысить производительность операции поиска узлов, а также помочь адаптировать иерархию октодерева к более широкому диапазону входных данных. Определены параметры формирования октодерева и методика их оценки, позволяющая принимать решения при разработке программного обеспечения.

Практическая ценность работы заключается в создании программной системы, подтверждающей теоретические результаты работы, и предназначенной для использования в процессе обработки облаков точек для ограничения потребления оперативной памяти. Предложенные подходы, методы и алгоритмы позволяют кратно снизить потребление оперативной памяти при выполнении типовых операций обработки облаков точек, сохраняя при этом приемлемую производительность.

**Обоснованность и достоверность** научных положений, основных выводов и результатов диссертации подтверждаются всесторонним анализом состояния исследований в данной области. Корректность предложенных методов и алгоритмов подтверждается согласованностью теоретических положений и результатов, полученных при практической реализации предложенных методов и алгоритмов, апробацией основных теоретических положений диссертации в печатных трудах и докладах, а также положительными результатами внедрения основных положений диссертации.

**Реализация результатов работы.**Представленные в работе методы и алгоритмы: создание растровых проекций больших облаков точек; поиск цилиндрических объектов, представленных трубопроводом, в облаке точек; гибридная обработка данных лазерного сканирования и фотосъемки для идентификации и атрибутирования точек лазерного сканирования были  разработаны в рамках выполнения работ по проекту «Исследование и разработка алгоритмов и программных средств по обработке, хранению и визуализации данных лазерного сканирования и фотосъемки» (Уникальный идентификатор проекта RFMEFI58417X0025) при финансовой поддержке Министерства науки и высшего образования Российской Федерации.

Апробация результатов работы. Результаты исследований докладывались на следующих конференциях: 19th International Conference on Computational Science and its Applications, Saint-Petersburg, 2019; XXI Международная конференция по мягким вычислениям и измерениям, Санкт-Петербург, 2018г.

Внедрение представленных алгоритмов и методов было произведено в рамках: проекта ФЦП «Исследования и разработки по приоритетным направлениям развития научно-технологического комплекса России на 2014 - 2020 годы" по теме: «Исследование и разработка алгоритмов и программных средств по обработке, хранению и визуализации данных лазерного сканирования и фотосъемки» (Уникальный идентификатор проекта RFMEFI58417X0025) индустриальным партнером ООО “Экоскан” (Соглашение о предоставлении субсидии от 03.10.2017 г. № 14.584.21.0025).

Зарегистрированы следующие результаты интеллектуальной деятельности (РИД): «Программа для создания растровых проекций», №2018665901, 11.12.2018; «Программа для сортировки и фильтрации облака точек», №2018666456, 17.12.2018; «Программное средство построения проекций облака точек, полученных от мобильного комплекса лазерного сканирования», № 2015617817, 22.07.2015.

Публикации. Автором опубликовано по теме диссертации 6 печатных работ, среди них 2 работы в рецензируемых журналах из перечня ВАК.

**Структура и объем работы**. Диссертационная работа объемом в 172 машинописных страницы состоит из введения, 4 разделов, списка литературы, списка сокращений и условных обозначений, списка из 62 иллюстраций и 11 таблиц, 14 листингов и предметного указателя.

ОСНОВНОЕ СОДЕРЖАНИЕ ДИССЕРТАЦИИ

Во введении содержится обоснование актуальности диссертационной работы, определены цели, объект и предмет исследований, а также методы исследований. Также сформированы основные научные результаты, выносимые автором на защиту, а также их практическая ценность.

В первой главе проводится анализ принципов работы систем лазерного сканирования, а также процессов получения, обработки и предобработки данных лазерного сканирования. Проводится обзор существующих методов обработки облаков точек в ограниченном объеме потребляемой оперативной памяти. Рассматриваются структуры разбиения пространства, и их применение для организации обработки облака точек с использованием вторичных систем хранения данных. Отмечаются ограничения существующих решений, формулируются основные задачи для дальнейшего исследования.

Лазерный сканер представляет собой систему дистанционного зондирования, позволяющую выполнять измерения расстояний до поверхности объекта сканирования, с высокой скоростью (от нескольких тысяч до миллиона измерений в секунду) и высокой точностью. Большинство современных моделей лазерных сканеров позволяют окрашивать облако точек, используя для этого встроенную цифровую фотокамеру. На текущий момент выделяют три вида лазерного сканирования: наземное лазерное сканирование (НЛС), мобильное лазерное сканирование (МЛС), воздушное лазерное сканирование (ВЛС). Различаются они преимущественно по методам привязки к глобальной системе координат, а также по точности и плотности финального облака точек.

Данные лазерного сканирования представлены облаком точек. Единицей информации в подобном облаке является точка. Помимо координаты в трехмерном пространстве, точка может (в зависимости от поддержки со стороны формата хранения) содержать следующие атрибуты:

* цвет;
* интенсивность;
* нормаль к поверхности сканирования;
* классификация;
* номер отражения;
* прочие параметры.

В **разделе 1.2** рассмотрены существующие форматы хранения облаков точек. Существующие форматы хранения облака точек делятся на открытые и проприетарные. Проприетарные используются в закрытом программном обеспечении (TerraSolid, Bentley и пр.), в силу чего не рассматриваются в данной работе. Основные открытые форматы хранения облака точек, в свою очередь, используются повсеместно, и поддерживаются практически всеми программами, работающими с облаком точек.

В диссертационной работе для хранения и загрузки облаков точек используется формат LAS, как наиболее распространенный и поддерживаемый. Другим положительным качеством данного формата является то, что при необходимости обеспечения более компактного представления облаков точек возможно использование родственного формата LAZ с минимальными изменениями в исходном коде.

В **разделе 1.4** рассмотрены проблемы пространственного поиска в облаках точек и применение структур разбиения пространства. Облако точек является неорганизованной структурой данных. В силу специфики лазерного сканирования, точки в облаке могут быть расположены рядом, в соответствии с траекторией луча сканирования. Однако, предсказать их распределение не представляется возможным: расположение точек в облаке может быть изменено произвольным образом в результате операций совмещения облаков точек, или в результате обработки.

Таким образом в облаке, представленном последовательным массивом точек, операции поиска в общем случае требуют полного перебора облака точек, то есть вычислительная сложность составляет (или в случае необходимости поиска нескольких точек). Такой подход может быть оправдан, когда количество операций поиска невелико, однако в случае применении операций поиска для каждой точки в облаке сложность подобной операции становится , что потребует применения более продвинутых способов организации пространственного поиска.

Для ускорения подобных вычислений обычно используются специальные структуры данных, описывающей объем, занимаемый облаком точек, в виде набора непересекающихся регионов, и позволяющая идентифицировать каждую точку в облаке как принадлежащую к определенному региону. При использовании подобной информации появляется возможность производить поиск с использованием информации о смежности регионов, а также использовать более дешевую проверку на соответствие критериям поиска для групп точек, что позволяет снизить сложность поиска до (или в случае необходимости поиска нескольких точек). Будем называть подобную структуру данных структурой разбиения пространства.

В **разделе 1.5** проведен анализ существующих подходов к обработке больших облаков точек. Детальное изучение существующих способов обработки облаков точек в условиях нехватки оперативной памяти позволяет выделить следующие подходы:

1. Сокращение объема загружаемых облаков точек за счет более компактного представления или сжатия. Может быть применен совместно с прочими. Не позволяет полностью избежать проблемы нехватки памяти.
2. Применение потоковой обработки облака точек. Позволяет решить проблему для алгоритмов, не использующих топологическую информацию облака точек. Путем введения предварительной сортировки точек позволяет использовать топологическую информацию в локальной зоне, не всегда содержащей все интересующие точки. Не позволяет выполнять отсечение не интересующих участков облака точек.
3. Выгрузка на вторичную систему хранения неактуальных на данный момент данных. Позволяет решить проблему за счет использования устройств большей емкости, однако вызывает усложнение архитектуры системы, и может привести к замедлению ее работы.

В данной работе производится анализ третьего способа: обработки больших облаков точек при ограничении потребляемой памяти за счет выгрузки неактуальных данных на вторичную систему хранения.

Современные подходы к решению такой проблемы используют методы разбиения пространства совместно с системой кеширования данных. Такой подход позволяет добиться большей локальности участков, чем кеширование неупорядоченного облака точек. Среди рассмотренных реализаций используются следующие структуры разбиения пространства: k-мерное дерево, тайловый массив, октодерево. К-мерное дерево относится к структурам разбиения, управляемым данными, в силу чего очень чувствительно к модификации облака точек, однако требует меньше памяти и обеспечивает более быстрый доступ к данным по сравнению с октодеревом. Тайловые или воксельные массивы, а также октодерево относятся к структурам разбиения, управляемым пространством, что позволяет не перестраивать структуру при изменениях в облаке точек. Кроме того, они лучше подходят для использования совместно с вторичной системой хранения, так как разбиение на каждом уровне идентично, что позволяет выполнять сохранение меньшего количества информации.

Также стоит отметить различия в механизмах регуляции потребления оперативной памяти. В системах, использующих выгрузку данных на вторичные системы хранения, преимущественно используется подход с организацией системы кеширования, так как она позволяют использовать динамическую подгрузку и выгрузку данных, позволяя при этом держать в оперативной памяти одновременно ограниченное количество данных. Однако, отдельного внимания заслуживает подход с использованием механизма отображения данных, позволяющий переложить эту работу на операционную систему, и предоставляющему доступ к данным по прямому указателю.

В большинстве подходов, в процессе построения создается большое количество файлов, что приводит к снижению производительности файловых операций. В некоторых файловых системах большое количество файлов может привести к исчерпанию доступного количества файловых записей в файловой системе.

В заключение главы произведена декомпозиция задачи «Анализ модели задач обработки, получение оценок сложности доступа к информации из облака точек, определение требований к предварительному структурированию информации с целью уменьшения сложности, определение критериев эффективности формируемых структур данных» на следующие подзадачи:

1. Исследование модели октодерева на базе системы кеширования.
2. Исследование модели октодерева на базе механизма отображения памяти.
3. Исследование методов компактного представления данных применительно к иерархической модели октодерева, подходящих для использования при работе в ограниченном объеме оперативной памяти.
4. Обеспечение минимального количества создаваемых файлов в октодереве для снижения нагрузки на файловую систему.

Во второй главе производится анализ модели задач обработки, получение оценок сложности доступа к информации из облака точек, определение требований к предварительному структурированию информации с целью уменьшения сложности, определение критериев эффективности формируемых структур данных. Для решения поставленной задачи предложены структуры данных и алгоритмы, предназначенные для построения октодерева, позволяющего выполнять обработку больших облаков точек c использованием вторичных систем хранения. Рассматриваются модели октодерева с использованием арифметики с плавающей точкой, целочисленного идентификатора узла. Предложена иерархическая модель с использованием целочисленной арифметики. Предложены способы организации работы во внешней памяти с использованием механизма кеширования и механизма отображения памяти.

В **разделе** **2.2** выполняется получение оценок сложности доступа к информации в облаке точек и октодереве. Облако точек— это неупорядоченное множество точек в трехмерном евклидовом пространстве, полученное в результате трехмерного сканирования объекта, и представляющее поверхность этого объекта:

Точке на поверхности сканируемого объекта может соответствовать не только трехмерная координата, но и другие параметры, полученные в результате сканирования или обработки облака точек. Такие параметры в дальнейшем будут называться *атрибутами* точки. Точка из облака точек представляет собой вектор (кортеж) координат (), нормалей (), цветов () и прочей информации, использующейся при построении облака точек:

где где – трехмерные координаты точки;

где – компоненты нормали;

*,* где *–* красная, зеленая и синяя компоненты цвета.

Проведена оценка сложности доступа к информации в облаке точек для операций вставки, удаления, получения и поиска элемента. Пусть облако точек представлено в виде динамического массива точек, последовательно расположенных в памяти.

При вставке элементов в облако точек различают две ситуации: вставка в середину и вставка в конец. Худшим случаем будет вставка в начало массива, что потребует перемещения всех элементов, что дает вычислительную сложность . При вставке элементов в конец массива нет необходимости производить перемещение элементов, однако при превышении емкости массива потребуется выделение нового участка памяти и перемещение в него всех данных облака точек. В таком случае сложность добавления элемента в конец составит , однако сложность добавления *n* элементов составит (при увеличении размеров согласно степени двойки). Таким образом, амортизированная сложность добавления элемента составляет .

Аналогично с операцией вставки, удаление элементов также требует перемещения всех последующих, что дает вычислительную сложность . Удаление последнего элемента имеет вычислительную сложность , так как не требует перемещения элементов массива.

Учитывая непрерывное расположение в памяти, местоположение элемента определяется его номером по порядку, что дает сложность доступа к элементу и сложность обхода всего массива .

Пространственный поиск в облаке точек без построения вспомогательных структур данных осуществляется при помощи полного перебора всех элементов. При поиске *k* ближайших элементов к определенной точке в пространстве вычислительная сложность поиска варьируется от (при ) до (при ).

Проведена оценка сложности доступа к информации в октодереве для операций разбиения узла, вставки, удаления и получения элемента. Октодерево является древовидной структурой, каждый узел которого содержит восемь потомков, а также содержит список точек, которые в него попадают.

Если при добавлении новой точки в узел он содержит максимально допустимое количество элементов, происходит его разбиение на восемь подузлов. В идеально сбалансированном октодереве, содержащем *n* листовых узлов, и имеющим глубину разбиения , будет узлов, а временная сложность полного разбиения октодерева, содержащего *n* точек составит , где *K* – стоимость обработки одной точки. Таким образом, амортизированная стоимость разбиения одного узла составит .

Добавление нового элемента в октодерево требует поиска подходящего узла для вставки. Поиск выполняется рекурсивно, начиная с корневого узла, и продолжается для его потомков до тех пор, пока не будет найден подходящий листовой узел. Учитывая то, что на каждом уровне разбиения размеры узла делятся пополам, процесс поиска подходящего узла будет иметь сложность ( в случае сильно несбалансированного дерева), а процесс добавления новой точки в узел будет иметь сложность .

Процесс удаления точек из октодерева выполняется за , однако процесс перестроения узлов октодерева может занимать . С учетом того, что процесс перестроения не выполняется после удаления каждой точки, можно сказать что удаление имеет временную сложность .

Получение элемента также требует поиска содержащего элемент узла (). Выборка элемента из массива точек узла имеет сложность , так как точки в таком массиве расположены в памяти непрерывно.

В **разделе 2.3** проведен анализ иерархической модели октодерева и способов кодирования узлов. Рассмотрены общая структура узла октодерева и способы ее компактного представления с использованием следующей методики:

1. Отказ от хранения центра и размеров узла, так как они могут быть вычислены в момент обхода дерева.
2. Хранение одного указателя на массив потомков вместо восьми указателей на узлы, а также введения 8 бит для идентификации существующих потомков.
3. Использование урезанных 6-байтовых указателей.
4. Сокращение размера точки, путем использования только двух байт на координату.

Рассмотрен способ представления иерархической структуры октодерева на арифметике с плавающей точкой. В таком случае узел будет задан тремя интервалами по каждой из координатных осей (пример для одномерного случая приведен на рисунке Рисунок 1). При этом координата и размеры корневого узла октодерева могут быть представлены любым значением с плавающей точкой.

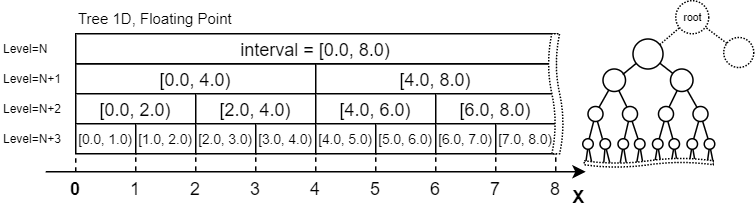


Рисунок - Фрагмент октодерева на арифметике с плавающей точкой

В подходе с плавающей точкой координаты и размеры узлов представлены floating-point значениями. Такое решение позволяет поддерживать более широкий диапазон входных значений. Однако, недостатком подобного подхода является снижение точности на слишком высоких и слишком малых величинах, а также повышенная сложность вычислений по сравнению с целочисленным подходом.

Рассмотрен способ представления иерархической структуры октодерева с использованием целочисленного идентификатора для кодирования местоположения узла в октодереве, который, при наличии информации о корневом узле, позволяет получить размеры и местоположение узла. При этом, иерархия октодерева остается аналогичной октодереву на арифметике с плавающей точкой, и остается возможность позиционирования октодерева в пространстве при помощи floating-point координат. Идентификатор представлен 64-разрядным целым (Рисунок 2), и сформирован следующим образом: для идентификации расположения узла выше или ниже плоскости разбиения по каждой из осей выделено по одному биту. Отличием является хранение информации по всем уровням разбиения, а также введение бита, идентифицирующего конечный уровень разбиения (leading one).

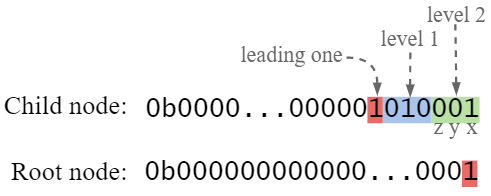


Рисунок - Идентификатор узла октодерева в бинарной форме

Рассмотрен способ представления иерархической структуры октодерева с использованием целочисленной арифметики. Используя рекурсивную природу октодерева можно обеспечить адресацию узлов в нем при помощи целочисленных координат, обеспечивающих однозначное сопоставление позиции узла в октодереве и его местоположение в пространстве. Размер узла в таком октодереве ограничен не сверху, как в подходе с плавающей точкой, а снизу, что позволяет наращивать размеры октодерева, указывая количество бит, выделенных на адресацию. Преимуществом такого подхода является возможность использования гибридного подхода в описании иерархии октодерева, используя младшие биты ключа для адресации в октодереве, и старшие для адресации для адресации участков пространства в иной системе хранения данных. Таким образом, в отличии от классической реализации октодерева, описанной ранее, появляется возможность расширения участка пространства, занимаемого октодеревом, по мере добавления новых данных.

В случае использования целочисленной арифметики местоположение узла будет задано битом или комбинацией бит в координате узла (пример для одномерного случая приведен на рисунке Рисунок 3). При этом координата и размеры корневого узла октодерева могут принимать только значения степени двойки.

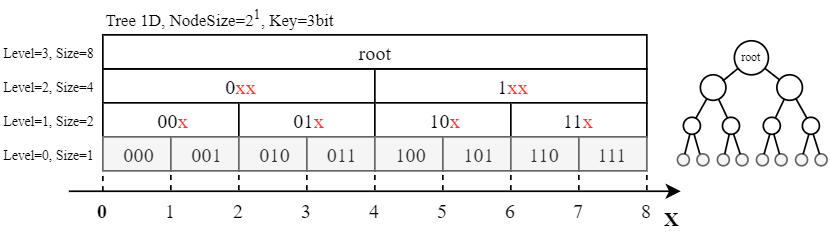


Рисунок - Дерево на арифметике с плавающей точкой

На рисунке Рисунок 4 приведен пример разбиения узла на нулевом уровне при различном количестве измерений и при различном количестве разбиений. Узел с параметрами *D=3* и *Dim=1* соответствует узлу октодерева.

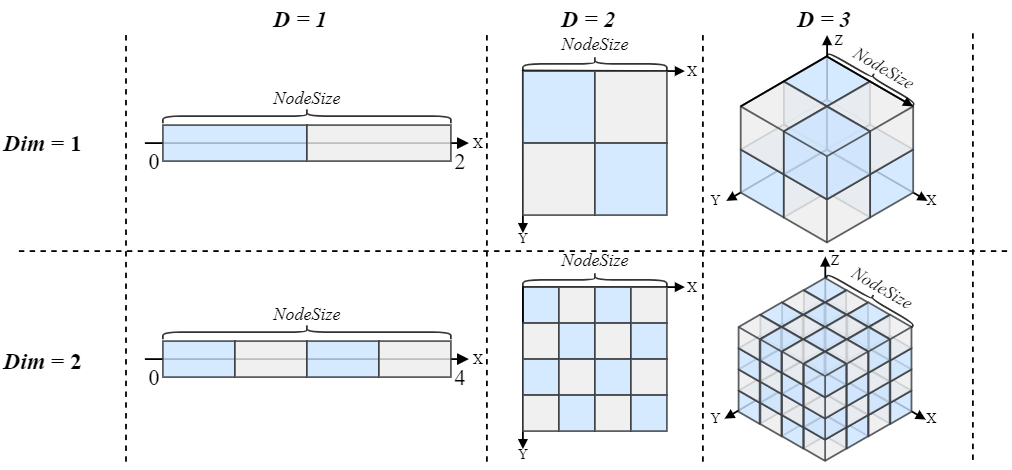


Рисунок - Пример расположения потомков узла в пространстве при различном количестве измерений

Рассмотрено применение ассоциативных массивов для хранения узлов целочисленного октодерева. Ассоциативные массивы выполняют хранение элементов в виде пары {ключ:значение}, и позволяют выполнять относительно быстрый поиск элементов по их ключам. Ассоциативные массивы плохо подходят для хранения точек, потому что не поддерживают операции пространственного поиска, и не имеют древовидной структуры, полезной при раннем отсечении участков облака. Однако, они могут быть использованы в совокупности с целочисленным октодеревом за счет использования первых бит координаты в качестве ключа в ассоциативном массиве (Рисунок 5).

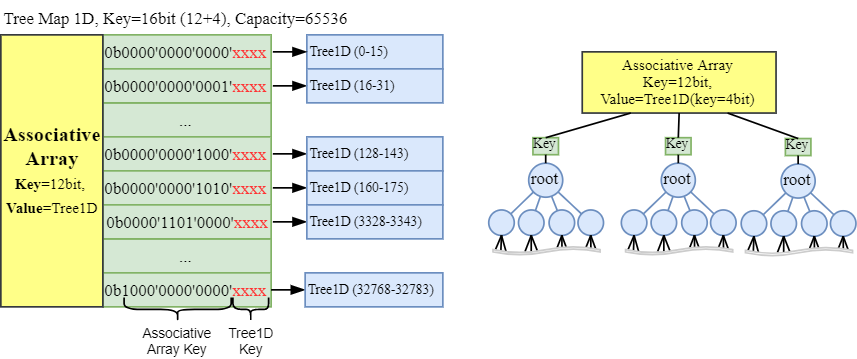


Рисунок - Интеграция целочисленного дерева с ассоциативным массивом

В **разделе 2.4** проведен анализ модели памяти октодерева в условиях ограниченного потребления оперативной памяти. В процессе анализа существующих методов работы в условиях ограничений оперативной памяти было предложено два метода управлением памятью:

* *Использование системы кеширования данных.* Данный метод выполняет управление размещением блоков данных при помощи двухуровневой системы кеширования, в которой первым уровнем является размещение блока в оперативной памяти, а вторым уровнем – размещение блока на жестком диске. При вытеснении блоков используется политика Least Recently Used (LRU).
* *Использование механизма отображения памяти.* Данный механизм позволяет выполнять отображение содержимого файла на диапазон виртуальных адресов процесса, что позволяет напрямую работать с сохраненными блоками точек как с данными в оперативной памяти.

На основании проведенного анализа модели управления потребляемой памятью для октодерева, предложен способ организации механизма асинхронного двухуровневого кеширования для октодерева, способ построения октодерева на базе системы кеширования, позволяющий снизить количество создаваемых файлов, способ организации октодерева с использованием механизма отображения памяти и алгоритмов динамической аллокации.

В третьей главе рассматривается реализация методов формирования октодеревьев на базе двухуровневой асинхронной системы кеширования и на базе механизма отображения памяти. Предложен алгоритм построения октодерева с использованием системы кеширования и возможностью объединения заполненных узлов в общий файл. Для обеспечения возможности использования механизма отображения памяти в октодереве предложена реализация алгоритма динамической аллокации на отображаемой памяти. Для обеспечения возможности обработки больших облаков точек сторонними библиотеками предложен способ обработки больших облаков точек путем интеграции системы аллокации отображаемой памяти в сторонние библиотеки для linux систем.

В **разделе 3.1** рассмотрен процесс загрузки облака точек и способы его ускорения. Рассмотрен алгоритм, позволяющий выполнить разделение задач загрузки участка облака точек из файловой системы и его предобработки на несколько потоков.

В **разделе 3.2** приводится описание алгоритма и структуры октодерева с применением асинхронной системы двухуровневого кеширования.

Адресация узла производится при помощи 64 битного беззнакового целого числа (uint64\_t), что позволяет производить адресовать 263 узла, или 21 уровень разбиения октодерева. Используя для адресации узла целочисленный идентификатор открывается возможность применения контейнеров, реализованных в стандартной библиотеке C++, далее называемой STL (Standard Template Library).

Для хранения списка узлов, загруженных в оперативную память используется двусвязный список. Для снижения количества операций выделения памяти используется собственная реализация двусвязного списка, позволяющая выполнять хранение пула освобожденных элементов и переиспользовать их при повторной аллокации.

Для подсчета занимаемой памяти в реальном времени используется механизм добавления к значению счетчика потребляемой памяти дельты в момент загрузки, создания, или удаления узла. Для поддержания заданного уровня потребления оперативной памяти *M*, необходимо в момент выделения новой памяти размера *N* (при создании узла или увеличении его данных) освободить узлы из конца двусвязного списка до обеспечения значения потребляемой памяти в ).

Реализация политики кеширования LRU выполняется при помощи предложенного двусвязного списка. Для обеспечения ее работы отслеживаются операции создания и загрузки узла, а также изменения его данных. При возникновении любой из таких ситуаций данный узел помещается в конец двусвязного списка. Учитывая, что освобождение элементов происходит с начала списка, часто используемые узлы будут оставаться в оперативной памяти дольше всего.

Асинхронное выполнение файловых операций реализовано в отдельном потоке, принимающем управляющие команды при помощи асинхронной неблокирующей очереди сообщений. Управляющая команда содержит тип выполняемой операции, указатель на данные облака точек, и индекс узла, для которого производится заявленная операция. Сформирован список команд, необходимый для взаимодействия со вторичной системой хранения.

Построение октодерева осуществляется путем последовательной вставки блоков точек, считанных из облака точек. Реализовано два варианта алгоритма построения октодерева – рекурсивный и параллельный. Рекурсивный алгоритм выполняет для каждого узла октодерева процедуру добавления дочек, при заполнении узла переводит его в состояние завершенного, а затем повторяет ту же процедуру уже для его потомков. Заполненные узлы при этом игнорируются. Параллельная реализация алгоритма выполняется так же, как рекурсивная, однако вместо рекурсивного вызова происходит добавления запроса в очередь выполнения, которая обрабатывается асинхронно при помощи пула потоков

Общая схема программной реализации октодерева на базе системы кеширования приведена на рисунке Рисунок 6. На рисунке изображен центральный компонент (Octree Interface), который предоставляет основные методы для выборки, обхода и построения октодерева. Для хранения узлов октодерева используется ассоциативный массив (Hash Table). Для хранения данных облака точек, находящихся на текущий момент в оперативной памяти, а также для обеспечения политики вытеснения при превышении установленного лимита потребления, используется двусвязный список (LRU Cache). Для взаимодействия с файловой системой применяется асинхронный подход, при котором все операции выполняются в отдельном потоке (File Thread), взаимодействие с которым происходит при помощи неблокирующей очереди сообщений (Async Queue).

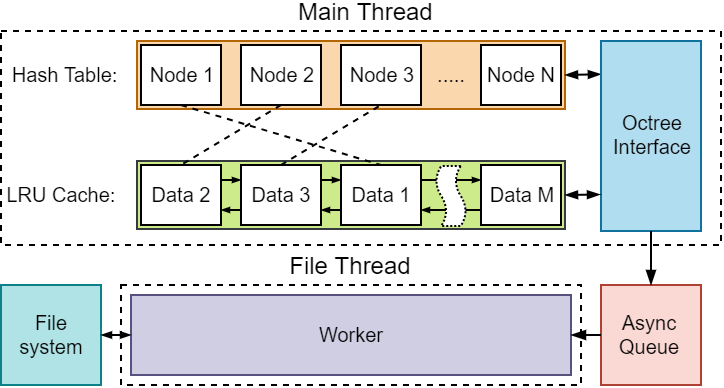


Рисунок - Общая схема программной реализации октодерева на базе системы кеширования

В **разделе 3.3** приводится описание алгоритма и структуры октодерева с применением механизма отображения памяти. Структура узла в данном октодереве существенно отличается от предложенной в октодереве на базе системы кеширования. С использованием механизма отображения памяти теперь не требуется введения громоздких механизмов кеширования, что существенно упрощает архитектуру октодерева.

Каждый узел октодерева выполняет хранение массива данных, который располагается в отображаемой памяти, механизмы работы с которым будут представлены в последующих разделах. Кроме массива данных узлом выполняется хранение указателя на массив потомков, а также битовые маски, указывающие на то, какие потомки не пусты. Каждый узел отвечает за представление диапазона целочисленных значений, который сужается по мере уменьшения его уровня. Имея целочисленную координату возможно получить узел, ее содержащий, на любом уровне разбиения. Младшие биты координаты отвечают за местоположение узла в целочисленном октодереве, а старшие за идентификацию в ассоциативном массиве.

Рассмотрено применение механизма отображения памяти для хранения данных узлов октодерева путем подмены аллокатора для реализации динамического массива в библиотеке STL. Подобный механизм позволяет переключаться на использование оперативной или отображаемой памяти без прочих изменений в исходном коде. Вместо системного аллокатора для хранения полезной нагрузки октодерева используется схема динамической аллокации на отображаемой памяти.

Общая схема программной реализации октодерева на базе механизма отображения памяти приведена на рисунке Рисунок 7. Иерархия октодерева представлена комбинацией классической организации узлов (Node) октодерева и ассоциативным массивом (Associative Array). Взаимодействие с октодеревом реализовано при помощи компонента (Octree Interface), содержащего основные методы доступа к данным. Данные узлов октодерева представлены динамическим массивом (Data), содержащим указатель на отображаемую память (Mapped Memory).

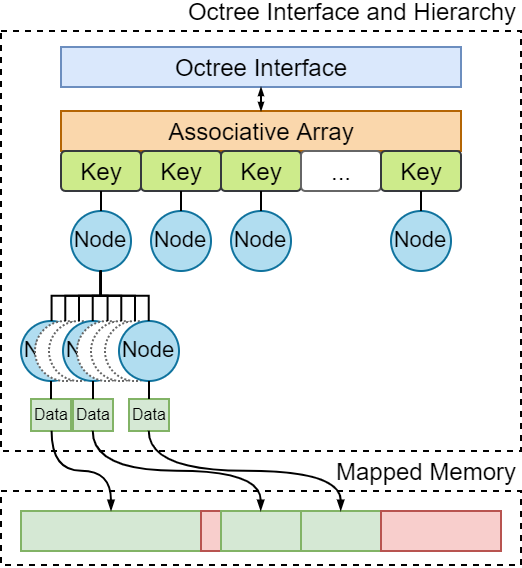


Рисунок - Общая схема программной реализации октодерева на базе механизма отображения памяти

В **разделе 3.5** рассмотрен способ обработки больших облаков точек путем интеграции системы аллокации отображаемой памяти в сторонние библиотеки для linux систем. Для обеспечения возможности обработки больших облаков точек была произведена интеграция библиотеки PCL с представленной в данной работе системы динамической аллокации на отображаемой памяти. В данном разделе рассмотрены алгоритмы, применяемые для решения задачи выделения цилиндрических объектов, а также способ их интеграции с системой динамической аллокации на отображаемой памяти для обеспечения возможности обработки больших облаков точек.

В четвертой главе проведено экспериментальное исследование предложенных октодеревьев, а также алгоритмов обработки облаков точек с использованием предложенных октодеревьев. Предложенные алгоритмы сравниваются с существующими решениями, рассматриваются их недостатки и ограничения.

В **разделе 4.2** произведен анализ эффективности октодеревьев при помощи измерения потребляемой памяти, времени построения и скорости поиска точек для облаков точек, приведенных в таблице Таблица 1 и на рисунке Рисунок 8. Выбор облаков точек был произведен руководствуясь критерием обеспечения дифференциации по размеру, плотности, топологии и методу съемки в целях демонстрации работы алгоритмов на широком спектре входных наборов данных. Представленные облака точек были получены в результате решения реальных промышленных и исследовательских задач.

Таблица - Облака точек для тестирования

|  |  |  |  |
| --- | --- | --- | --- |
| № | Название | Кол-во точек, млн | Размер, ГБ |
| 1 | five.las | 1.2 | 0.03 |
| 2 | polytech.las | 51.6 | 1.8 |
| 3 | smolny.las | 126 | 4.3 |
| 4 | molodezhnoe.las | 1560 | 53.4 |

|  |  |
| --- | --- |
|  |  |
|  |  |

Рисунок - Внешний вид используемых облаков точек с раскраской по плотности

Произведено измерение скорости построения октодерева в зависимости от параметров максимального количества точек в узле, и объемов кеша (Рисунок 9).

|  |  |
| --- | --- |
| D:\Nextcloud\disser2019\materials\charts\artone2\Test_results_-_Build___Node_Size.png | D:\Nextcloud\disser2019\materials\stdoctreetest\new3\Octree-_Build_summary.png |
| Рисунок - Сравнение скорости построения октодерева с кешированием (слева) и октодерева с отображением (справа)  Произведено сравнение скорости поиска в октодереве в зависимости от параметров максимального количества точек в узле, и объемов кеша (Рисунок 10). | |

|  |  |
| --- | --- |
| D:\Nextcloud\disser2019\materials\charts\artone2\Test_results_-_Search___Node_Size.png | D:\Nextcloud\disser2019\materials\stdoctreetest\new3\Octree-_Query_summary.png |
| Рисунок - Сравнение скорости поиска в октодереве с кешированием (слева) и октодереве с отображением (справа) | |

Установлено, что при использовании октодеревьев занимаемая облаком точек память оставалась в пределах заданных ограничений, увеличение объемов кеша положительно влияет на производительность операций построения и поиска, увеличение максимального количества точек в узле положительно влияет на скорость построения октодерева и скорость поиска узлов, однако отрицательно влияет на скорость поиска индивидуальных точек.

На основании анализа оценки влияния параметров октодеревьев на производительность операций построения и поиска была сформирована методика выбора параметров октодерева для разработки программного обеспечения.

В **разделе 4.3** произведено сравнение с существующими библиотеками и программами (Таблица 2), предназначенными для обработки и визуализации облаков точек с целью доказательства того, что предложенная система позволяет производить обработку больших облаков точек в ограниченном объеме оперативной памяти, без существенного ущерба в производительности по сравнению с системами, выполняющими работу полностью в оперативной памяти.

Таблица - Сравнение скорости построения и потребляемой памяти для различных реализаций октодерева

|  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| № | Размер облака, ГБ | Октодерево на базе системы кеширования | | Октодерево  на базе отображения памяти | | Point Cloud Library | | Cloud Compare | |
| Время, сек | Память, ГБ | Время, сек | Память, ГБ | Время, сек | Память, ГБ | Время, сек | Память, ГБ |
| 1 | 0.03 | 0.06 | 0.0334 | 0.13 | 0.0015 | 0.122 | 0.0066 | 0.435 | 0.114 |
| 2 | 1.8 | 2.6 | 0.517 | 3.8 | 0.004 | 4.6 | 1.8 | 18.2 | 1.6 |
| 3 | 4.3 | 11.6 | 0.586 | 10 | 0.014 | 12.5 | 4.3 | 58.5 | 3.6 |
| 4 | 53.4 | 894 | 0.986 | 950 | 0.180 | - | - | - | - |

На основании проведенного экспериментального анализа параметров предложенных октодеревьев установлено, что:

* Увеличение объемов кеша положительно влияет на производительность операций построения и поиска.
* Увеличение максимального количества точек в узле положительно влияет на скорость построения октодерева и скорость поиска узлов, однако отрицательно влияет на скорость поиска индивидуальных точек.

В **разделе 4.4** рассмотрено применение октодерева для решения задачи построения растровых проекций. Результат построения приведен на рисунке Рисунок 11.

|  |  |
| --- | --- |
|  |  |

Рисунок – Визуализация данных внутри октодерева (слева) и результат растровой проекции (справа)

В **разделе 4.6** произведена экспериментальная оценка алгоритма выделения цилиндрических объектов на базе механизма отображения памяти. Эксперимент был поставлен путем сравнения скорости выполнения и потребления памяти на различных этапах алгоритма при работе с использованием стандартного системного аллокатора (т.е. в оперативной памяти) и с использованием аллокатора на отображаемой памяти (т.е. в файловой памяти). В качестве облака точек для тестирования было выбрано molodezhnoe.las, размером 53 ГБ.



Рисунок - Потребление файловой памяти при работе алгоритма

Было установлено, что при использовании отображаемой памяти потребление алгоритмом оперативной памяти составило 16 мегабайт, при этом все основные объемы данных были выделены в отображаемой памяти. Полученные результаты позволяют сделать вывод, что предложенный метод позволяет выполнять обработку больших облаков точек используя алгоритмы, рассчитанные на применение исключительно в оперативной памяти, без существенного снижения производительности.

В **заключении** сформулированы основные итоги, представленные теоретическими и практическими результатами и выводами, полученными в ходе диссертационного исследования.

ЗАКЛЮЧЕНИЕ

В работе решена задача по выполнению обработки больших облаков точек в ограниченном объеме оперативной памяти при помощи октодеревьев, использующих вторичную систему хранения. Был применен системный подход для выявления компонентов системы обработки больших облаков точек, определения связей между ними, их функций и целей, и механизмов управления ресурсами системы и объектно-ориентированный подход для создания необходимых абстракций и интерфейсов в процессе реализации. Совокупность разработанной модели, методов и алгоритмов, а также их практическая реализация представляют собой решение актуальной научно-технической задачи обработки больших облаков точек при условии наличия ограничений на потребление оперативной памяти, и позволяет снизить требования к вычислительным ресурсам системы. В ходе решения данной задачи были получены следующие результаты:

1. В результате исследования существующих алгоритмов обработки больших облаков точек, отмечено преимущественное использование структур разбиения пространства для группировки близко расположенных точек, и использование системы кеширования для поддерживания заданного объема потребления оперативной памяти. Отмечена узкая специализация существующих алгоритмов и тенденция к созданию большого количества файлов. Обоснован выбор октодерева в качестве используемой структуры разбиения, а также сформированы требования к такой структуре.
2. На основании проведенного исследования моделей задач обработки, полученных оценок сложности доступа к информации из облака точек, приведенных требований к предварительному структурированию информации с целью уменьшения сложности а также рассмотренных способов компактного представления узла октодерева с целью обеспечения обработки больших облаков точек был разработан метод формирования октодерева на базе двухуровневой асинхронной системы кеширования и метод формирования октодерева с использованием механизма отображения памяти.
3. Разработаны иерархические структуры данных октодерева, включающие реализацию на целочисленной арифметике и на базе целочисленного идентификатора. Разработаны алгоритмы формирования октодерева и системы управления потребляемой памятью, включающие реализацию на базе двухуровневой системы кеширования и реализацию на базе механизма отображения памяти в связке с динамическим аллокатором.
4. Разработано программное обеспечение для выполнения обработки облаков точек ЛС с применением предложенных алгоритмов. Предложен способ обработки больших облаков точек путем интеграции системы аллокации отображаемой памяти в сторонние библиотеки для linux систем и разработано программное обеспечение для обработки облаков точек при помощи сторонней библиотеки в интеграции с предложенными алгоритмами.
5. Разработано программное обеспечение для экспериментального исследования обработки облаков точек при помощи предложенных алгоритмов и структур разбиения. Предложена система показателей, позволяющая оценить производительность операций построения и поиска в зависимости от различных параметров октодерева. Разработана методика выбора параметров октодерева, необходимых для разработки программного обеспечения.

**Рекомендации и перспективы дальнейшей разработки темы.** Совершенствование иерархической структуры октодерева с целью уменьшения ее объема и увеличения производительности поиска, а также обеспечения адаптации для большего количества вариаций входных данных, совершенствование механизмов хранения данных с целью ускорения процедур выборки и сохранения данных.

Соответствие паспорту специальности. Данное диссертационное исследование выполнено в соответствии с паспортом специальности 05.13.01 «Системный анализ, управление и обработка информации (технические системы)», а именно соответствует следующим областям (номера соответствуют пунктам в паспорте специальности): п. 4 – Разработка методов и алгоритмов решения задач системного анализа, оптимизации, управления, принятия решений и обработки информации; п. 12 – Визуализация, трансформация и анализ информации на основе компьютерных методов обработки информации.

Список публикаций по теме диссертации

**ВАК:**

1. Беляевский К.О. Формирование октодерева по облаку точек при ограничении объема оперативной памяти // НТВ СПбГПУ. Информатика. Телекоммуникации. Управление. 2019. Т. 4.
2. Badenko V., Tammsaar S., Beliaevskii K., Fedotov A., Vinogradov K. Multithreading in Laser Scanning Data Processing // Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics) 11619 LNCS, 2019. Pp. 289-305.

**РИНЦ:**

1. Баденко В.Л., Беляевский К.О., Волгин Д.Ю., Зотов Д.К., Федотов А.А. Гибридный подход к 3D реконструкции сложных индустриальных объектов из облака точек // Мягкие вычисления и измерения: сборник трудов XXI Международной конференции, 2018. Т. 1. С. 308-311.
2. Беляевский К.О., Болсуновская М.В. Использование целочисленной арифметики для формирования октодерева // Неделя науки СПбПУ. 2019. Т. 1. С. 123-125.
3. Беляевский К.О., Болсуновская М.В. Использование механизма отображения памяти при формировании октодерева облака точек // Неделя науки СПбПУ. 2019. Т. 1. С. 126-128.

**Scopus:**

1. Bolsunovskaya M., Tammsaar S., Beliaevskii K., Fedotov A., Gintciak A. Experimental sample of a software module for processing of a cloud of laser scanning points for natural-technical systems development // Proceedings of the 33rd International Business Information Management Association Conference, IBIMA 2019: Education Excellence and Innovation Management through Vision 2020, 2019. Pp. 8619-8627.

**Результаты интеллектуальной деятельности:**

1. Таммсаар С.В. Программа для создания растровых проекций / С.В. Таммсаар, К.О. Беляевский, Н.С. Чупин. – Свидетельство о государственной регистрации программы для ЭВМ № 2018665901 от 11.12.2018.
2. Таммсаар С.В. Программа для сортировки и фильтрации облака точек / С.В. Таммсаар, К.О. Беляевский, Н.С. Чупин. – Свидетельство о государственной регистрации программы для ЭВМ № 2018666456 от 17.12.2018.
3. Болсуновская М.В. Программа хранения облака точек лазерного сканирования для природно-технических систем / М.В. Болсуновская, А.В. Лексашов, А.М. Гинцяк, К.О. Беляевский. – Свидетельство о государственной регистрации программы для ЭВМ № 2019660673 от 09.08.2019.
4. Болсуновская М.В. Программа визуализации облака точек лазерного сканирования для природно-технических систем / М.В. Болсуновская, А.В. Лексашов, А.М. Гинцяк, К.О. Беляевский. – Свидетельство о государственной регистрации программы для ЭВМ № 2019619801 от 24.07.2019.
5. Таммсаар С.В. Программное средство построения проекций облака точек, полученных от мобильного комплекса лазерного сканирования / С.В. Таммсаар, К.О. Беляевский. – Свидетельство о государственной регистрации программы для ЭВМ № 2015617817 от 22.07.2015.
6. Беляевский К.О. Программа для гибридной обработки облака точек / К.О. Беляевский, С.В. Таммсаар. – Свидетельство о государственной регистрации программы для ЭВМ № 2019665354 от 22.11.2019.
7. Беляевский К.О. Программа для визуализации облака точек / К.О. Беляевский, С.В. Таммсаар. – Свидетельство о государственной регистрации программы для ЭВМ № 2019665556 от 25.11.2019.