

Московский государственный университет имени М.В. Ломоносова

Факультет вычислительной математики и кибернетики

Кафедра суперкомпьютеров и квантовой информатики

Арбузов Николай Романович

**Исследование эффективности использования kd-деревьев при параллельной обработке облаков точек**

ВЫПУСКНАЯ КВАЛИФИКАЦИОННАЯ РАБОТА

**Научный руководитель:**

к.ф.-м.н., доцент

И. М. Никольский

Москва, 2023

Оглавление

[Введение 3](#_Toc132380952)

[Обзор литературы 4](#_Toc132380953)

[Постановка задачи 6](#_Toc132380954)

[Параллельное kd-дерево 7](#_Toc132380955)

[Алгоритм поиска 9](#_Toc132380956)

[Результаты исследования 11](#_Toc132380957)

[Список литературы 13](#_Toc132380958)

# Введение

Обработка облаков точек — это важная задача, которая имеет широкое применение в таких областях, как компьютерное зрение, графика, геодезия и многих других. Обработка этих данных требует высокой вычислительной мощности, поскольку облака точек могут содержать миллионы или даже миллиарды точек. Одним из наиболее эффективных способов индексации облаков точек является использование kd-деревьев, которые позволяют выполнять операции поиска, добавления и удаления точек с высокой скоростью и эффективностью. Однако, большинство существующих алгоритмов обработки облаков точек с использованием kd-деревьев являются последовательными и не могут полностью использовать мощность параллельных вычислений. Это ограничение означает, что время, необходимое для обработки больших объемов данных, может быть значительно увеличено, что делает такие задачи более сложными и затратными.

Целью данного исследования является создание и разработка параллельного алгоритма индексирования облаков точек с помощью kd-дерева, а также исследование его эффективности в условиях параллельных вычислений. Для достижения данной цели были поставлены следующие задачи:

* Изучение существующих работ по использованию kd-деревьев в обработке облаков точек и параллельных алгоритмов обработки облаков точек
* Разработка методики исследования, включающей описание используемых данных и методов обработки облаков точек, а также алгоритма параллельной обработки облаков точек с использованием kd-деревьев. Планируется создать алгоритм индексации данных, распределенных по узлам суперкомпьютера
* Проведение экспериментального исследования для сравнения эффективности параллельной обработки облаков точек с использованием kd-деревьев и последовательной
* Анализ результатов экспериментов и оценка эффективности использования kd-деревьев в параллельной обработке облаков точек.

Результаты данного исследования могут быть использованы для улучшения существующих алгоритмов обработки облаков точек с использованием kd-деревьев и разработки новых параллельных алгоритмов.

# Обзор литературы

Про облака точек

С того момента, как в 1975 году Джон Бентли создал и описал принципы работы kd-дерева (Bentley J., 1975), многие ученые различными способами улучшали методы применения данной структуры, например, используя приближенные вычисления с помощью Akd-дерева (Approximate kd-дерева), в котором для решения задачи ICP (Iterative Closest Point, то есть алгоритм поиска ближайшего соседа) ищется не действительно ближайший сосед, а ближайшая точка из того блока, в котором располагается заданная точка (Greenspan M., Yurick M., 2003). Однако, у такого подхода есть явно следующий из названия минус: в большинстве случаев данный алгоритм будет выдавать результат только с некоторой ошибкой, в то время как классическое kd-дерево дает точный результат. Другим примером улучшения может служить поиск в kd-дереве с кэшированием (Cashed kd-tree search) суть улучшения состоит в том, что узлы помимо указателей на свои дочерние узлы так же хранят и указатель на родительский узел, корневой узел хранит нулевой указатель, таким образом время поиска уменьшается вплоть до 50%, из-за перехода на уровень выше во время возможных тестов ball-within-bounds, тестов на поиск возможных ближайших точек в соседей узлах (Nuchter A., Lingemann K., Hertzberg J., 2007).

В работе с пространственными данными часто используются kd-деревья, так как имеют архитектуру, заточенную именно под них. Среди областей применения можно выделить такие, как трассировка лучей в реальном времени с использованием обычных процессоров (Shevtsov M., Soupikov A., Kapustin A., 2007) и с использованием графических ускорителей (Zhou K. et al., 2008). Помимо трассировки лучей, kd-деревья могут также использоваться и при обработке облаков точек, например, полученных с помощью датчиков LiDAR (Zhou H. et al., 2021).

Как известно, помимо kd-деревьев для индексации пространственных данных часто используются такие структуры, как октодерево, r-дерево и так далее. Октодерево – это древовидная структура данных, узлы которой окружают некоторую кубическую секцию пространства и либо являются листьями, либо имеют по 8 потомков (Meagher D., 1980). В отличии от kd-деревьев октодеревья часто бывает несбалансированным ввиду своей идеи выделать сразу по 8 дочерних узлов при необходимости. Вследствие этого, как описано в работе Адамсона и Воркапика (Adamsson M., Vorkapic A., 2016), октодеревья занимают в несколько раз больше места в оперативной памяти и затрачивают в несколько раз больше времени на выполнения поиска. Если брать r-деревья, то они представляют из себя древовидную структуру, узлы которой являются вложенными прямоугольниками (параллелепипедами), окруженными корневым узлом (Guttman A., 1984). Однако, как описано в работе Нарасимхулу (Narasimhulu Y. et al., 2021), r-дерево так же уступает kd-дереву по временным показателям, это происходит из-за того, что как алгоритм построения kd-дерева, так и алгоритмы поиска в нем проще и требует меньших вычислительных мощностей по нескольким параметрам: kd-дерево – это, в отличие от r-дерева, бинарное дерево, а значит каждый узел имеет максимум 2 дочерних узла, в то время, как в r-дереве их может быть больше.

Выше было подробно описано kd-дерево и приведены его улучшения и работы с его использованием, теперь обратимся к параллельности при обработке облаков точек.

# Постановка задачи

В качестве выборки для проведения тестов и измерений будут использованы облака точек, приведенные на Рисунке 1, а также на облаках точек в виде кубов с 100 000, 1 000 000 и 10 000 000 точек, заполненных случайным образом. При выполнении данной работы будет измеряться время построения kd-дерева и время выполнения функции поиска точек на этом дереве в заданном диапазоне для сравнения последовательной и параллельной программы и измерения эффективности параллельной версии.

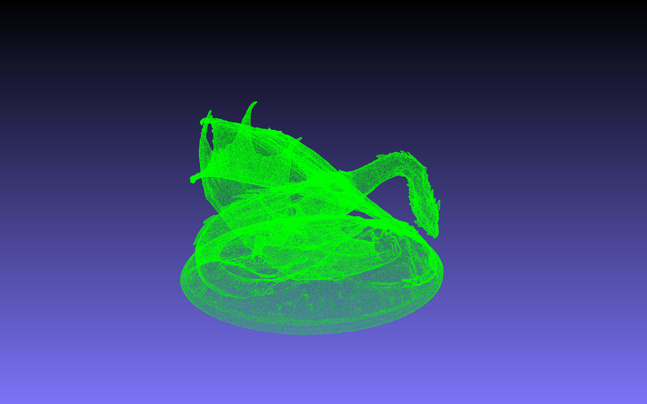


Рисунок . Слева: облако точек Fantasy Dragon (2 499 971 точек, https://www.artec3d.com/3d-models/fantasy-dragon). Справа: облако точек Bearded guy (1 024 356 точек, https://www.artec3d.com/3d-models/bearded-guy-hd)

Параллельность алгоритма будет основана на использовании парадигмы MPI (Message Passing Interface), она предоставляет набор функций для передачи сообщений между процессами, а также для синхронизации выполнения процессов.

Для проведения опытов будут использованы локальный компьютер в части тестовых запусков и визуализации облаков, а также параллельная вычислительная система Polus для основных измерений. Локальный компьютер оснащен 4‑ядерным процессором Intel Core i5 с частотой 2 ГГц и ОЗУ объемом 16 Гб с частотой 3733 МГц. Система Polus состоит из 5 вычислительных узлов, на каждом узле установлены по 2 10-ядерных процессора IBM POWER8 и каждое ядро имеет 8 потоков, также Polus имеет общую оперативную память 256 Гбайт (в узле 5 оперативная память 1024 Гбайт) с ЕСС контролем.

# Параллельное kd-дерево

В качестве модели для изучения была создана модель распределенного kd-дерева, в котором процессоры выступают в роли узлов (Рисунок 2). В этом дереве данные на не листовых узлах хранятся только до тех пор, пока не будут переданы дальше. Таким образом не происходит дублирования данных, ведь они хранятся только на листовых узлах.

0

2

………

i

2i+1

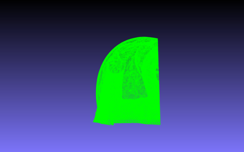
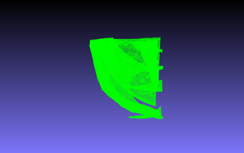
2i+2

1

Рисунок . Схема распределения процессов по дереву.

Построение распределенного kd-дерева (Рисунок 3) мало чем отличается от построения локального и происходит следующим образом: процесс с нулевым рангом считывает облако точек, определяет в нем среднюю точку по оси и индексом 0, допустим это будет ось X. Далее этот процесс передает, если это возможно, своему левому потомку точки, у которых координата X меньше координаты найденной средней точки и, соответственно, правому потомку остальные точки. На следующем уровне все происходит аналогично за исключением координаты, по которой производятся сравнения: для процессов с рангами 1 и 2 сравнения будут производиться по оси с индексом 1, допустим это ось Y, а для следующего уровня (процессы с рангами 3, 4, 5 и 6) по оси с индексом 2, допустим это ось Z. Таким же образом облако точек распределяется и для более глубоких уровней. Ось можно определить с помощью ранга по формуле:

Таким образом на листовых узлах появляются части облака точек, по которым строятся локальные kd-деревья.



Proc. 2



Proc. 1

Proc. 0

Proc. 3

Proc. 4

Proc. 5



Рисунок . Организация распределенного дерева и передачи частей облаков точек в нем на примере системы с 5 процессами.

# Алгоритм поиска

В качестве алгоритма, на котором будет проверяться эффективность разработанной структуры, был выбран алгоритм поиска точек внутри заданной области пространства (параллелепипед) (Рисунок 4).

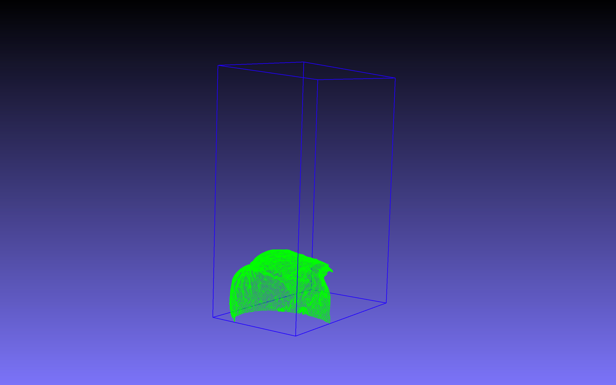
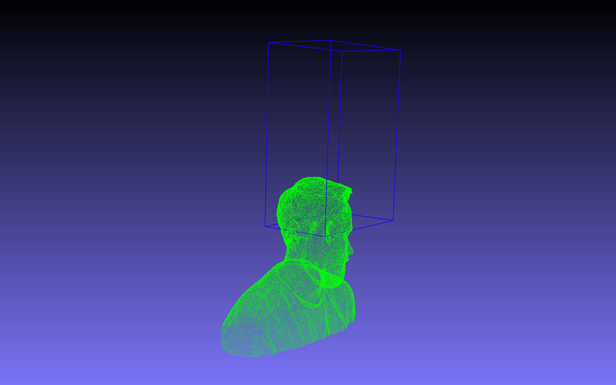
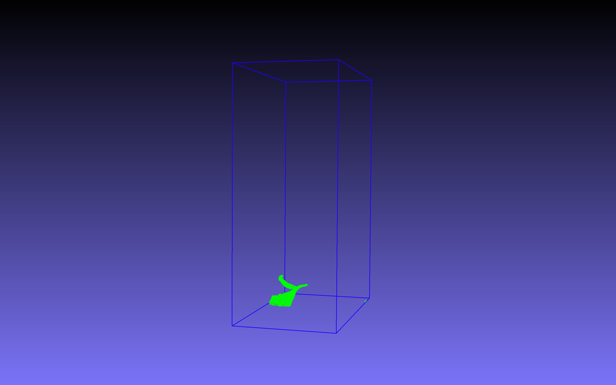
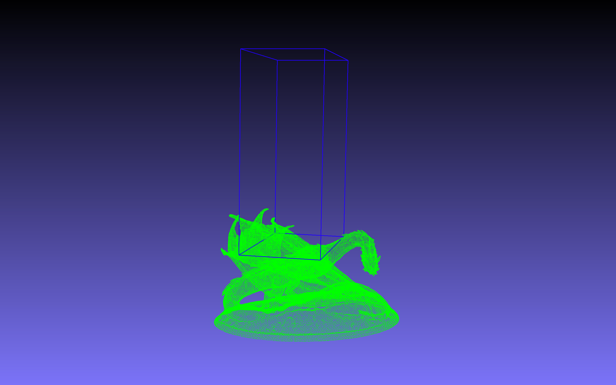


Рисунок . Пример работы алгоритма поиска точек в заданном пространстве для облаков точек: Fantasy Dragon и Bearded guy.

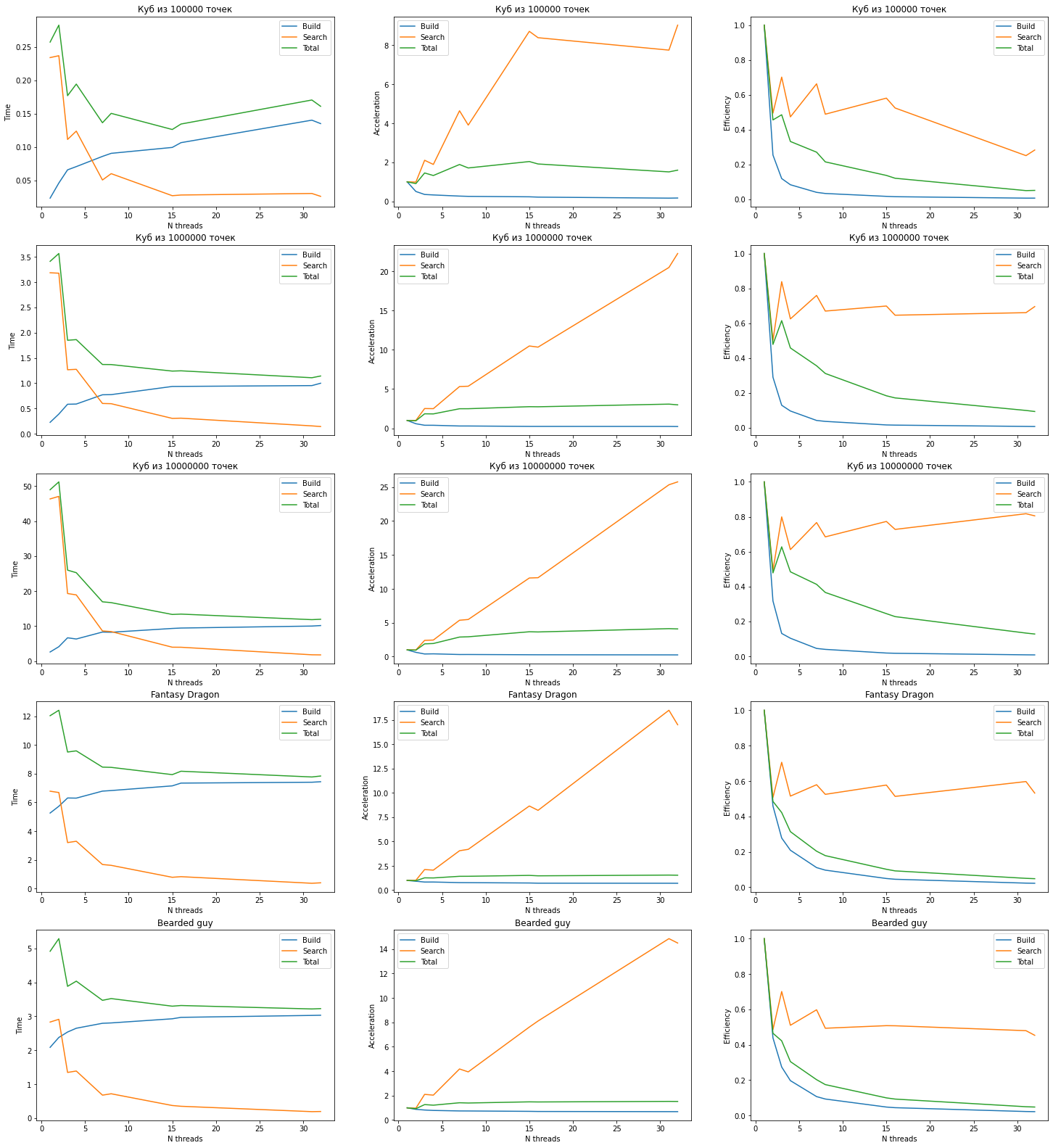
В процессе с рангом 0 определяются границы области: случайным образом, вводятся пользователем или определяются постоянными для одинаковых условий во время экспериментов. Далее эти границы передаются до листовых узлов, если хотя бы часть данных, содержащихся в этих листовых узлах, лежит внутри границ, иначе этим узлам передаются одинаковые границы пространства (пространства с нулевым объемом), это обозначает, что необходимых данных в них нет и от них не будет ожидаться передача найденных точек. После того как листовые узлы получили границы поиска, они производят поиск точек внутри этих границ точек своего локальном kd-дерева и, получив список таких точек, отправляют его родительскому узлу, те, в свою очередь, получают данные от всех своих потомков, объединяют их и отправляют своим родительским узлам. Так происходит до тех пор, пока все найденные точки не будут получены корневым узлом (процессом с рангом 0). Таким образом процесс с рангом 0 получает полный набор точек из облака, содержащийся внутри заданной области пространства может применять его для дальнейшей работы.

Данный алгоритм может быть использован во многих областях, где используются и облака точек, при небольшом количестве доработок. Например, можно преобразовать алгоритм для поиска множества точек в произвольной области пространства (не только в параллелепипеде) и использовать найденное множество для локальных преобразований: фильтрация выбросов или триангуляция облака точек, данные преобразования будут выполнятся гораздо быстрее, так как обработка будет происходить лишь на части облака точек. Из более прикладных задач можно выделить такие задачи, как выделение области облака точек для дальнейшего уточнения: увеличение плотности облака точек с помощью специальных алгоритмов. После такой операции уточненный фрагмент облака точек можно эффективнее использовать в автоматическом управлении транспортными средствами или в медицине.

# Результаты исследования

На рисунке 5 приведены графики и таблицы результатов экспериментов, проведенных на системе Polus. По можно сделать несколько выводов:

Рисунок . Графики (справа налево): время выполнения, ускорение и эффективность построения дерева (синим), поиска точек (оранжевым) и общего времени (зеленым) в зависимости от количества процессов для облаков точек (сверху-вниз): куб с 100 000 точек, куб с 1 000 000 точек, куб с 10 000 000 точек, облако точек Fantasy Dragon и облако точек Bearded guy.



* Как и следовало ожидать алгоритм построения распределенного kd-дерева замедляется при увеличении числа процессов. Это происходит из-за увеличения числа пересылок данных.
* С другой стороны, алгоритм поиска точек в заданной области пространства дает заметное ускорение. Это особенно заметно на облаках с большим количеством точек (куб из 10 000 000 точек против куба из 100 000 точек), на них, в отличии от облаков с небольшим количеством, с увеличением числа процессов, эффективность остается близкой к постоянной (от 0,55 до 0,8 в зависимости от облака точек).
* Чем больше точек находится в облаке, тем более эффективно работает не только алгоритм поиска, но и вся программа в целом, это можно заметить, сравнивая результаты работы программы на облаках точек куб с 100 000 точек и куб с 10 000 000 точек.
* Так же можно видеть, что в случае, когда на дереве процессов полностью заполнен самый глубокий слой (3, 7, 15 и т. д. процессов) эффективность алгоритма поиска точек резко повышается.

# Основные результаты

Итак, в ходе работы были выполнены все поставленные задачи:

* Были изучены работы по созданию (Bentley J., 1975), улучшению (Greenspan M., Yurick M., 2003) и применению kd-деревьев при решении задач обработки пространственных данных, в том числе и облаков точек (Shevtsov M., Soupikov A., Kapustin A., 2007; Zhou K. et al., 2008). Так же были изучены работы по сравнению эффективности работы kd-деревьев с подобными структурами, такими как октодеревья (Adamsson M., Vorkapic A., 2016) и r-деревья (Narasimhulu Y. et al., 2021).
* Были описаны облака точек и алгоритмы, используемые в работе. А также создан действующий алгоритм распределенного облака точек.

# Список литературы

**Adamsson M., Vorkapic A.** A comparison study of Kd­tree, Vp­tree and Octree for storing neuronal morphology data with respect to performance [Отчет] : Degree Project in Computer Science / Computer Science and Communication ; KTH Royal Institute of Technology. - Stockholm : [б.н.], 2016.

**Bentley J.** Multidimensional Binary Search Trees Used for Associative Searching [Дневник] // Communications of the ACM. - New YorkNYUnited States : Association for Computing Machinery, 1 сентябрь 1975 г.. - 9 : Т. 18. - стр. 509–517.

**Greenspan M., Yurick M.** Approximate kd tree search for efficient ICP [Конференция] // Proceedings of the Fourth International Conference on 3-D Digital Imaging and Modeling (3DIM 2003). - Kingston, Ontario, Canada : IEEE, 2003. - стр. 442 - 448.

**Guttman A.** R-trees: a dynamic index structure for spatial searching [Конференция] // Proceedings of the 1984 ACM SIGMOD international conference on Management of data. - Berkeley : University of California , 1984. - стр. 47–57.

**Meagher D.** Octree Encoding: A New Technique for the Representation, Manipulation and Display of Arbitrary 3-D Objects by Computer [Книга]. - Troy : Rensselaer Polytechnic Institute, Image Processing Laboratory, 1980. - Т. 1.

**Narasimhulu Y. et al.** CKD-Tree: An Improved KD-Tree Construction Algorithm [Конференция] // Proceedings of the ISIC 2021: International Semantic Intelligence Conference. - New Delhi, India : SCIS, University of Hyderabad, 2021. - стр. 211-218.

**Nuchter A., Lingemann K., Hertzberg J.** Cached kd tree search for ICP algorithms [Конференция] // Proceedings of the Sixth International Conference on 3-D Digital Imaging and Modeling (3DIM 2007). - Osnabruck, Germany : IEEE, 2007. - стр. 419-426.

**Shevtsov M., Soupikov A., Kapustin A.** Highly parallel fast KD‐tree construction for interactive ray tracing of dynamic scenes [Дневник] // Computer Graphics Forum. - Oxford, UK : Blackwell Publishing Ltd, 2007 г.. - 3 : Т. 26. - стр. 395-404.

**Zhou H. et al.** Research on volume prediction of single tree canopy based on three-dimensional (3D) LiDAR and clustering segmentation [Дневник] // International Journal of Remote Sensing. - 2021 г.. - 2 : Т. 42. - стр. 738-755.

**Zhou K. et al.** Real-time kd-tree construction on graphics hardware [Дневник]. - [б.м.] : ACM Transactions on Graphics (TOG), 2008 г.. - 5 : Т. 27.