

Санкт-Петербургский государственный университет

Можеев Павел Евгеньевич

Выпускная квалификационная работа

Разработка воксельного SLAM-алгоритма с использованием плоскостей

Уровень образования: бакалавриат

Направление *02.03.03 «Математическое обеспечение и администрирование
информационных систем»*

Основная образовательная программа *СВ.5162.2020 «Технологии программирования»*

Научный руководитель:
старший преподаватель, к.т.н. Ю. В. Литвинов

Консультант:
инженер-исследователь, Mobile Robotics Lab, Skoltech А. В. Корнилова

Рецензент:
научный сотрудник, Технический университет Мюнхена, к.ф.-м.н. А. В. Артемов

Санкт-Петербург
2024

Saint Petersburg State University

Mokeev Pavel

Bachelor's Thesis

Voxel-based planar SLAM pipeline

Education level: bachelor

Speciality *02.03.03 "Software and Administration of Information Systems"*

Programme *CB.5162.2020 "Programming Technologies"*

Scientific supervisor:
C.Sc., senior lecturer Y. V. Litvinov

Consultant:
Research engineer at Mobile Robotics Lab, Skoltech A. V. Kornilova

Reviewer:
C.Sc., research scientist at Technical University of Munich A. V. Artemov

Saint Petersburg
2024

Оглавление

Введение	4
1. Постановка цели и задач	6
2. Обзор	7
2.1. Способы распознавания плоскостей	7
2.2. SLAM-алгоритмы, использующие плоскости	12
2.3. Наборы данных для оценки SLAM-алгоритмов	15
2.4. Метрики точности работы SLAM-алгоритмов	16
2.5. Инструменты для вычисления метрик	18
2.6. Вывод	19
3. Библиотека с воксельными SLAM-алгоритмами	20
3.1. Архитектура библиотеки	20
3.2. Реализация библиотеки	22
4. Экспериментальное исследование	24
4.1. Цель и вопросы эксперимента	24
4.2. Условия эксперимента	24
4.3. Исследование SLAM-алгоритмов	27
4.4. Вывод	31
5. Заключение	33
Список литературы	34
Приложение А	45

Введение

Задачи по определению текущего местоположения, а также геометрии окружающей среды являются ключевыми для автономных систем, ставших полноценной частью повседневной жизни человека. Для эффективного решения этих задач и смежных с ними проблем используются специализированные SLAM-алгоритмы. SLAM-алгоритмы (*Simultaneous Localization And Mapping*) — это алгоритмы, решающие две тесно связанные и дополняющие друг друга задачи: локализация и построение карты. Задача локализации состоит в определении местоположения автономной системы на основе данных, полученных с определенного типа сенсоров. На этапе построения карты выполняется дополнение и уточнение новыми извлеченными ориентирами карты, построенной на ранних итерациях работы системы. Концепция SLAM-алгоритмов заключается в том, что два этих этапа работают одновременно и зависят друг от друга: для лучшего определения местоположения нужна карта, и для большей точности карты необходимо понимание позиции автономной системы в пространстве. Оба этапа сводятся к задаче многомерной оптимизации и последующей оценке вектора состояний, содержащего положения автономной системы и ориентиров в среде.

Для непрерывной и эффективной работы SLAM-алгоритмы декомпозируются на две части (рис. 1): фронтенд — извлекает ориентиры из данных, полученных с определенного вида сенсоров, а также отвечает за построение ассоциаций между ориентирами и наблюдениями; бэкенд — вычисляет местоположение автономной системы в пространстве в каждый момент времени, а также занимается оценкой и обновлением карты, состоящей из извлеченных ориентиров.

Источниками данных для SLAM-алгоритмов могут служить стереокамера, камера глубины или лидар (LiDAR - англ. *Light Detection and Ranging*), позволяющий собирать трехмерные координаты точек (*облака точек*) объектов вокруг всей автономной системы. В качестве извлекаемого ориентира из полученных данных могут использоваться: ци-

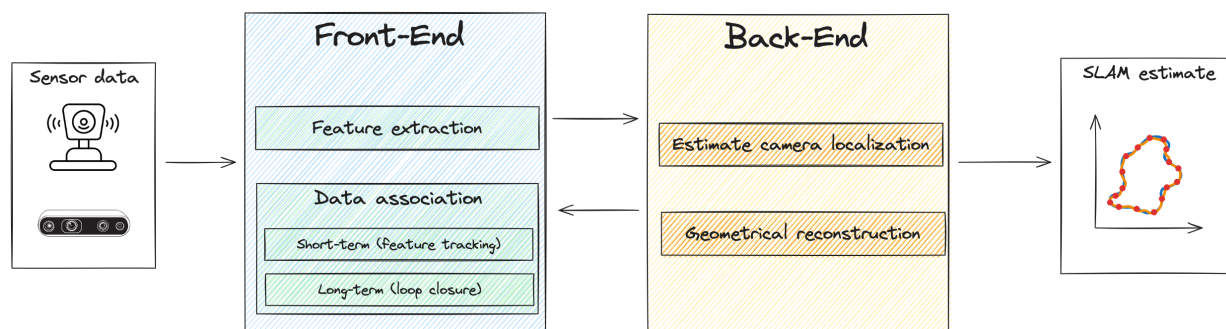


Рис. 1: Компоненты SLAM алгоритмов

линдры, сферы или плоскости. Плоскости, или планарный тип примитивов, являются наиболее популярным типом ориентиров за счет большой распространенности в рукотворной среде (стены, заборы, дорога, бордюры), а также возможности их комбинирования с другими видами примитивов. Использование плоскостей в качестве извлекаемого ориентира, а также лидара в качестве источника данных позволяет добиться лучшего понимания архитектуры окружающей среды и большей гибкости в экспериментах с SLAM-алгоритмами.

Для извлечения плоскостей из облаков точек, полученных с лидаров, существует несколько классов алгоритмов, основанных на: применении различных математических моделей [27, 74], принципах кластеризации [39, 78], постепенного увеличения размера группы точек [19, 84], использовании кубов (*вокселей*) [16, 71]. Последний тип алгоритмов выделяется из этой группы в силу превосходящих результатов, согласно исследованиям [2, 16, 47], большей предсказуемости и повторяемости (в сравнение с алгоритмами, основанными на моделях и алгоритмах), меньших вычислительных затрат (в сравнении с подходами, использующими принципы кластеризации). Несмотря на все преимущества данного подхода, реализации существующих SLAM-алгоритмов грубо привязаны к определенному исследованию и не предоставляют возможности реализации новых стратегий, что не позволяет сделать их работу модульной и легко воспроизводимой.

Целью данной работы является реализация адаптивного и модульного воксельного SLAM-алгоритма с использованием плоскостей.

1. Постановка цели и задач

Цель работы — реализовать модульный воксельный SLAM-алгоритм с использованием плоскостей и сравнить его с существующими аналогами.

Для достижения данной цели были поставлены следующие задачи.

1. Провести обзор существующих подходов к извлечению плоскостей из облаков точек, решений планарной SLAM-задачи, метрик, оценивающих результаты их работы, наборов данных, на которых проводятся эксперименты.
2. Спроектировать и разработать Python-библиотеку, позволяющую реализовывать и поддерживать различные компоненты воксельных SLAM-алгоритмов с использованием плоскостей для запуска различных стратегий.
3. Провести экспериментальное исследование различных стратегий воксельных SLAM-алгоритмов с использованием плоскостей.
4. Провести экспериментальное сравнение разработанных стратегий воксельных SLAM-алгоритмов с существующими решениями SLAM-задачи.

2. Обзор

В данном разделе рассмотрены способы извлечения плоскостей из облаков точек, полученных с лидара, существующие SLAM-алгоритмы, наборы данных, на которых проводятся эксперименты, метрики, оценивающие результат работы SLAM-алгоритмов, а также инструменты, благодаря которым можно посчитать значения данных метрик.

2.1. Способы распознавания плоскостей

Для решения задачи извлечения плоскостей из облаков точек, полученных с лидара, существует два класса алгоритмов (рис. 2). Первый класс таких методов (*point-based*) работает напрямую с точками, полученными с сенсоров. Данный класс в свою очередь делится на три подхода, основанных на: использовании различных математических моделей аппроксимации плоскости (*model-fitting-based*); кластеризации точек (*clustering-based*); постепенном увеличении грубо оцененной группы точек на основе схожих вычисляемых параметров (*region-growing-based*). Второй класс алгоритмов (*voxel-based*) для извлечения плоскостей из облаков точек оперирует точками, находящимися в соответствующих кубах, по которым они распределяются на первоначальных этапах работы подобных методов. В данном разделе рассмотрен каждый из вышеупомянутых подходов.

2.1.1. Подход, основанный на использовании математических моделей

Подход, основанный на использовании математических моделей аппроксимации плоскости, заключается в их итеративном применении для извлечения ориентира из облаков точек. Двумя самыми популярными представителями данного подхода являются RANSAC и преобразование Хафа.

Идея алгоритма RANSAC [21] (*RANdom SAmple Consensus*) заключается в первоначальной оценке плоскости на случайной группе то-

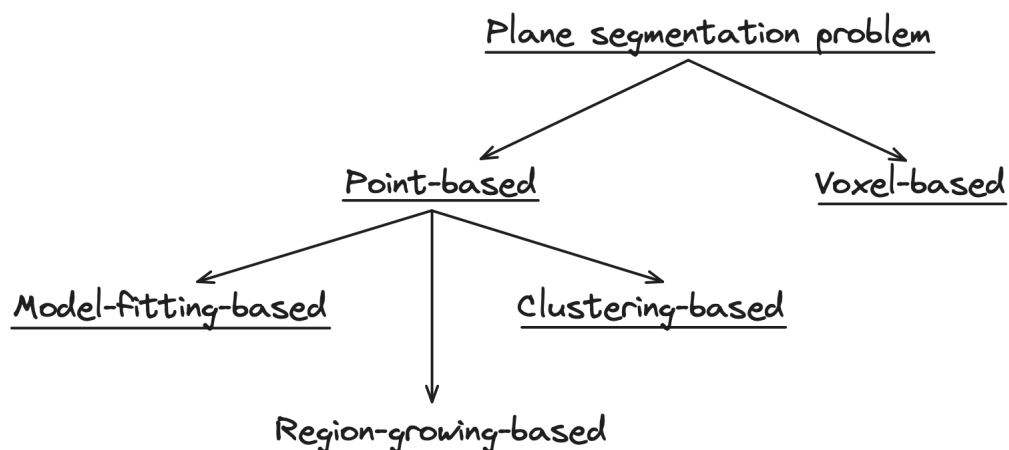


Рис. 2: Алгоритмы извлечения плоскостей из облаков точек

чек с последующим добавлением к ним тех, которые находятся на не превышающем заданный порог расстоянии. После выполнения данных шагов RANSAC выбирает плоскость с наилучшим результатом метрики, реализованной алгоритмом. Параметрами данного метода являются количество первоначальных точек, на основе которых будет оценена первоначальная плоскость, и пороговое значение (*threshold*), превышая которое точка будет считаться «внешней» для плоскости. Данный алгоритм является одним из фундаментальных в мире SLAM-задачи и имеет множество оптимизаций вычислений в силу их тяжеловесности [10, 11, 33, 42, 62].

Идея преобразования Хафа [5, 29] заключается в обработке каждой точки и вычислении набора параметров, зависящего от искомого ориентира, с последующим выбором лучшего результата (*голосование*). Данный метод позволяет помимо плоскостей [1] извлекать также цилиндры [57] и сферы [63]. Последовательная обработка точек сказывается на производительности данного метода, вследствие чего были предложены реализации вероятностного [35, 43, 64] и случайного [72] преобразования Хафа, позволяющие проводить этап голосования лишь на определенном множестве точек.

К плюсам данного класса методов можно отнести надежность в контексте того, что даже из зашумленных участков данных вышеописанные алгоритмы способны извлекать ориентиры. К минусам описанного

подхода относятся тяжеловесность вычислений, а также сильная чувствительность к данным, на которых запускается метод.

2.1.2. Подход, основанный на кластеризации

Идея методов, основанных на кластеризации, заключается в группировке точек по определенным параметрам без выбора опорного элемента для старта работы алгоритма. Данные алгоритмы могут начать работу с произвольного места в полученных с лидара данных, разрастаясь и группируя точки со схожими параметрами. Нюансом работы подобных методов могут стать точки на границе карты, ввиду чего зачастую они исключаются из первоначального рассмотрения и уточняются определенными эвристиками на финальных этапах работы алгоритмов. Условием добавления точки в существующие кластеры могут быть различные параметры, так, например, авторы статьи [50] используют расстояние от кластера до точки, исследования [8, 25] применяют количество соседей у границы кластера, а работа [30] — расстояние и угол наклона от начала координат до кластера.

Преимуществом методов, работающих по данному принципу, является возможность извлекать ориентиры из облаков точек без проведения какой-либо предварительной работы, как, например, выбор опорного элемента для кластеризации на основе вычисляемых параметров, что занимает дополнительное процессорное время и ресурсы. К недостаткам описанного подхода можно отнести требовательность к вычислительным ресурсам, а также сильную зависимость результата работы алгоритма по извлечению ориентиров из данных от их зашумленности и граничных значений.

2.1.3. Подход, основанный на увеличении первоначальной группы точек

Принцип работы алгоритмов, основанных на увеличении первоначальной группы точек, схож с методами, основанными на кластеризации. Их ключевое различие заключается в том, что первый тип алго-

ритмов на первом шаге находит наилучшие точки (*семена*) для старта кластеризации, после чего добавляют к ним другие ориентиры, подходящие под вычисляемые условия. Впервые идеи данного подхода были предложены в статье [6], где авторы используют в качестве точки старта кластеризации набор пикселей, удовлетворяющих заданному значению кривизны поверхности. В дальнейшем идеи этой статьи получили развитие в работах [58, 68]. Современные алгоритмы, использующие рассматриваемый подход, в качестве условия для продолжения увеличения области точек зачастую используют: нормали плоскости [14] или остаточные значения [58], угол [34] или расстояние [26, 66] между семенами. Описанный подход может применяться не только обособленно, но и в сочетании с другими алгоритмами, например, после выделения кластеров со схожими признаками (воспользовавшись методом кластеризации), можно провести их объединение по определенным параметрам.

Плюсом данного подхода является воспроизводимость, так как набор данных и параметров остается постоянным. К минусам можно отнести зависимость качества работы алгоритма от выбора начальных ориентиров и параметров для увеличения области, помимо этого данные алгоритмы достаточно требовательны к вычислительным ресурсам.

2.1.4. Подход, основанный на использовании группы точек в форме кубов

Работа данных алгоритмов заключается в делении облака точек на кубы (*воксели*) и хранение их в соответствующей структуре данных, после чего полученные воксели ассоциируются между различными наблюдениями. В качестве структуры данных для хранения вокселей используют октодеревья [32, 47, 83] или k -мерные деревья [7]. В контексте трехмерных данных наибольшей популярностью пользуются октодеревья за счет большей интуитивности представления данных рассматриваемой размерности и эффективности операций, достигаемой с помощью иерархической структуры.

Ключевым механизмом работы с вокселями является возможность

их деления на более мелкие (октеты), в зависимости от вычисляемого значения планарности текущей группы точек, что позволяет даже из зашумленных участков данных извлекать определенные ориентиры. В качестве условий деления вокселей можно выделить: значение минимального собственного числа обычной [2, 12, 32, 38, 53, 59, 87], вероятностной [16, 71] или взвешенной [60] матрицы ковариации или его соотношения с максимальным [54], рассчитанным с использованием Principal Components Analysis по группе точек¹; стандартное отклонение [12, 83]; минимальный размер вокселя [9, 47]; глубина октодеревя [38, 59]; максимальное [47, 65] или среднее [53] количество точек в вокселе.

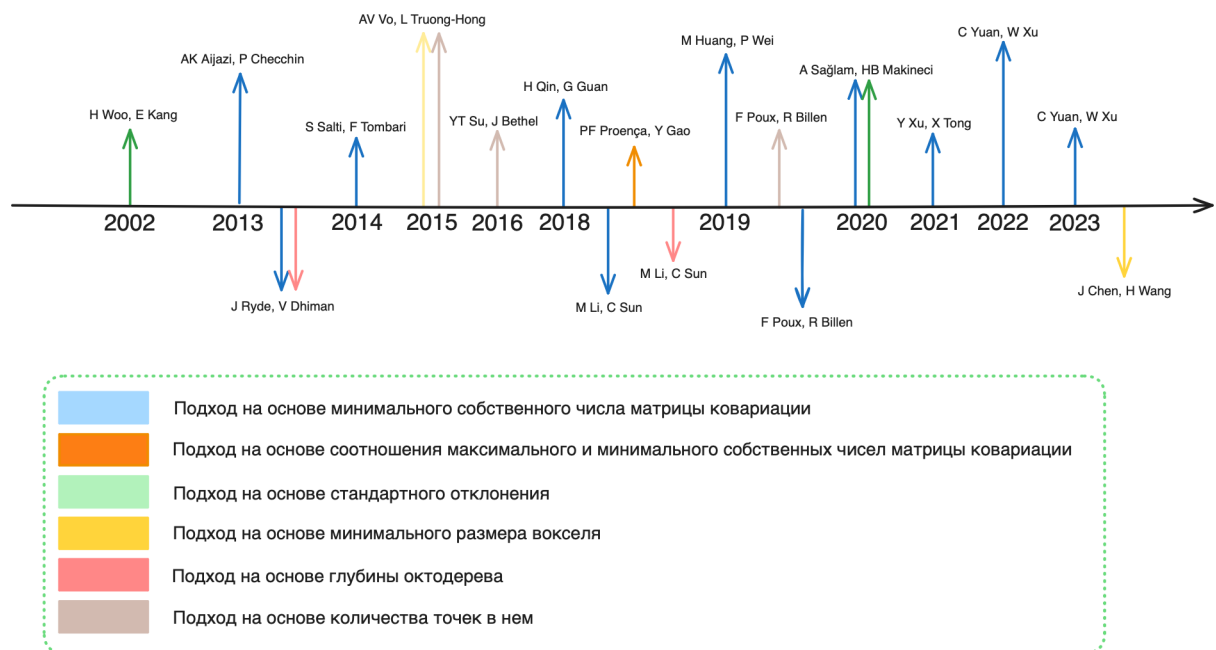


Рис. 3: Хронологический обзор подходов деления вокселей

На рисунке 3 представлен хронологический обзор подходов деления вокселей. Из данного графика видно, что наиболее популярным и часто применяемым условием дальнейшего деления куба с точками является использование минимального собственного числа матрицы ковариации. Данный подход стал особенно популярным в последнее время ввиду исследований, использующих вероятностную составляющую при обработке облаков точек.

¹<https://builtin.com/data-science/step-step-explanation-principal-component-analysis>
(дата обращения: 01.05.2024)

Преимуществами данного подхода является воспроизводимость и повторяемость результатов, а также точность при извлечении плоскостей из зашумленных участков облаков точек. К возможным недостаткам можно отнести требовательность к оперативной памяти, что может быть решено за счет грамотного использования структуры данных октодерева.

2.2. SLAM-алгоритмы, использующие плоскости

За историю существования SLAM-задачи появилось немало способов решить ее с использованием плоскостей. В данном разделе будут рассмотрены самые популярные из подобных алгоритмов, а также их компоненты. Спецификации и подходы рассматриваемых SLAM-алгоритмов структурированы в таблице 1.

Исследование	Способ извлечения плоскостей	Тип бэкенда
A. Trevor; J. Rogers [67]	Model-fitting-based	ICM
H Zhang, C Ye [74]	Model-fitting-based	ICM
Y Taguchi, YD Jian [52]	Model-fitting-based	ICM
WS Grant, RC Voorhies [27]	Model-fitting-based	ICM
L Zhou, S Wang [78]	Hybrid	ICM
L Zhou, D Koppel [77]	Hybrid	ICM
E Mendes, P Koch [45]	Clustering-based	ICP
Z Liu, X Liu [39]	Clustering-based	ICM
K Lenac, A Kitanov [19]	Region-growing-based	ICM
R Dubé, A Gawel [84]	Region-growing-based	ICM
C Yuan, W Xu [16]	Voxel-based	ICM
Y Yuan, C Wu [71]	Voxel-based	ICM
J Chen, H Wang [9]	Voxel-based	ICM
K Ćwian, MR Nowicki [37]	Voxel-based	ICM

Таблица 1: Используемые компоненты в SLAM-алгоритмах

Авторы статей [52, 67] были одними из первых, кто стал извлекать плоскости из облаков точек в качестве ориентиров для решения планарной SLAM-задачи. В дальнейшем идеи использования планарных ориентиров для SLAM-алгоритмов получили развитие, например, в статьях [27, 74], авторы которых воспользовались алгоритмами

RANSAC и преобразованием Хафа соответственно. Исследователи в статьях [39, 45, 78] решают SLAM-задачу также с использованием плоскостей в качестве извлекаемого ориентира, но применяют для этого подход, основанный на кластеризации точек. Методы, идеей которых является постепенное увеличение первоначальной грубо оцененной группы точек, находят применение в исследованиях [19, 84]. А результаты статей [16, 37, 71] основываются на алгоритмах, использующих воксели с распределенными по ним точками для последующих операций. Недостатком всех перечисленных выше реализаций фронтенд части воксельных SLAM-алгоритмов является грубая привязка к определенному исследованию, что затрудняет дополнять их новыми стратегиями, а также выполнять сравнительный анализ.

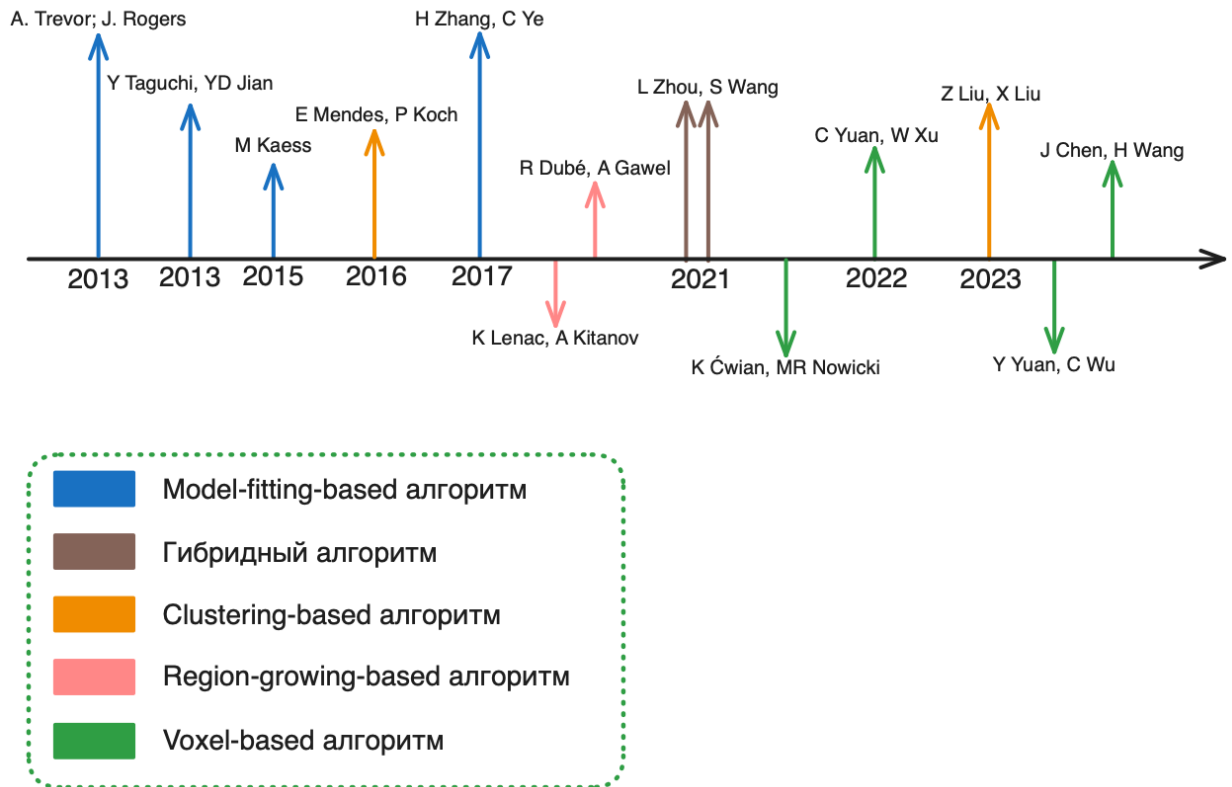


Рис. 4: Хронологический обзор SLAM-алгоритмов

Для вычисления и последующей оптимизации положения автономной системы в пространстве алгоритмы, описанные выше, используют различные бэкенды, основанные на принципе минимизации вычисляемой ошибки положения облаков точек. Одним из примитивных спосо-

$$\begin{aligned}
ICP_{point-to-point} &= \min \sum ||y_i - \bar{x}_i||^2 \\
ICP_{point-to-plane} &= \min \sum ((y_i - \bar{x}_i) * n_i)^2
\end{aligned}$$

Рис. 5: Функция ошибки метода Iterative Closest Point, где y_i , x_i точки, а n_i — нормаль плоскости.

$$\begin{aligned}
Q &= S_1 + T_2 S_2 T_2^T + \dots + T_H S_H T_H^T \\
\min_{\pi} (\pi^T Q \pi) &= \lambda_{\min}(Q)
\end{aligned}$$

Рис. 6: Функция ошибки метода ICM, где S_i — облако точек, T_i — матрица его трансформации, π — уравнение плоскости, наблюдаемой в различные моменты времени.

быв решения данной задачи является метод итерации ближайших точек (*Iterative Closest Point*) [75, 76], для которого существуют различные функции ошибки (рис. 5), как, например, точка-к-точке (*point-to-point*), рассматривающий точки как отдельные сущности и минимизирующий расстояние между их группами; точка-к-плоскости (*point-to-plane*), рассчитывающий соответствующую ошибку путем проецирования точки на другую группу точек, рассматриваемую как плоскость. Данный способ хоть и является эффективным, но требует больших вычислительных затрат.

Еще одним механизмом, благодаря которому достигается эффективная оптимизация положений объектов на карте, является использование интегрированной матрицы стоимости (*Integrated Cost Matrix*). Идея данного метода заключается в минимизации минимального собственного числа ICM матрицы, зависящей от облаков точек и матриц их трансформации (рис. 6). Данный метод получил развитие во многих работах [20, 48, 77, 78] и пользуется популярностью в силу точности и эффективности вычислений. Идеи данного подхода нашли свое практическое применение в библиотеках: FAST_LIO [73], FAST_LIO2 [18], LLOL [41], BALM [40, 39], mrob [81], SDSF [17] и MVE [22].

2.3. Наборы данных для оценки SLAM-алгоритмов

На текущий день, появилось достаточное количество наборов данных, позволяющих проводить всесторонние эксперименты над SLAM-алгоритмами. Рассмотрим основные из используемых в исследованиях наборы данных:

- **KITTI** [23] — набор данных, содержащий в себе 20 сцен с открытым пространством, для половины из которых есть калибровочные файлы и файлы с траекториями лидара, полученные с использованием GPS. Помимо лазерных данных, содержит также RGB-снимки.
- **Hilti** [82] — это несколько наборов данных (за 2021, 2022, 2023 года), который предоставляет Hilti Group, для соревнования разработанных SLAM-алгоритмов. Наборы данных содержат как облака точек с открытых пространств, так и с внутренних помещений зданий, к части из которых есть калибровочные данные и файлы с траекториями лидара.
- **NuScenes** [49] — набор данных, содержащий в себе 1000 сцен продолжительностью 20 секунд каждая. Представлен в формате некоторой реляционной базы данных. Помимо лазерных данных, указанный набор данных содержит снимки с шести камер, а также пяти радаров.
- **Durlar** [15] — набор данных, полученных со 128-лазерного (канального) лидара, что позволяет оперировать более чистыми облаками точек, по сравнению с лидарами меньшей канальности. Калибровочные и траекторные данные к снимкам также предоставлены, ввиду использования GNSS/INS-системы.
- **Waymo** [61] — набор данных, полученный с использованием пяти лидаров, размещенных на крыше и боковых сторонах автомобиля. Набор данных покрывает большую площадь территорий с открытым пространством и содержит 1950 сцен продолжительностью

Набор данных	Лидар	Количество каналов	Точность (см)	Точность эталонных XYZ позиций лидара (см)
KITTI	Velodyne HDL-64E	64	± 2	10
Hilti (2021)	Ouster OS0-64	64	± 1	0.3
Hilti (2022, 2023)	Hesai PandarXT-32	32	± 1	1
NuScenes	Velodyne HDL32E	32	± 2	2
Durlar	Ouster OS1-128	128	± 1	150
Waymo	Конфиденциально ²	-	-	-
Newer	Ouster OS0-128	128	± 1	3

Таблица 2: Используемые в наборах данных лидары

20 секунд каждая. К полученным с лидаров снимкам есть файлы с траекториями лидара.

- **Newer College Dataset** [46] — набор данных, полученный в ходе пешей прогулки по кампусу университета New College. Полученные снимки содержат такие артефакты, как быстрое передвижение лидара, а также его тряску. Для полученных облаков точек с лидаров есть файлы с траекториями передвижения.

Более подробные конфигурации лидаров, используемые в описанных наборах данных, указаны в таблице 2.

2.4. Метрики точности работы SLAM-алгоритмов

Ввиду того, что результатом работы SLAM-алгоритмов являются положения автономной системы и объектов на карте в различные моменты времени в глобальных координатах, способ оценивания качества работы подобных методов зависит от предоставленных данных и делится на два класса: референсный и безреференсный. Референсный подход используется в тех случаях, когда у последовательности рассматриваемого набора данных существует эталонная (*референсная*) траектория движения лидара, ввиду чего полученные с помощью оптимизаций траектории и положения объектов можно оценить не только качественно (визуально), но и численно с использованием метрик:

- Absolute Trajectory Error (*абсолютная ошибка траектории*) [44,

²Источник: <https://github.com/waymo-research/waymo-open-dataset/issues/544> (дата обращения: 01.05.2024).

69] — метрика, показывающая абсолютную разницу между референсной и оптимизированный траекториями лидара:

$$E_i := P_{est,i}^{-1} P_{ref,i}$$

$$ATE = \left(\frac{1}{n} \sum_{i=1}^n ||trans(E_i)||^2 \right)^{\frac{1}{2}}$$

где $P_{ref,i}, P_{est,i} \in SE(3)$ — референсная и оптимизированная траектории лидара соответственно.

- Relative Pose Error (*относительная ошибка положения*) [44, 69] — метрика, показывающая разницу между референсным и оптимизированным положениями лидара:

$$F_i^\delta := \left(P_{ref,i}^{-1} P_{ref,i+\delta} \right)^{-1} \left(P_{est,i}^{-1} P_{est,i+\delta} \right)$$

$$RPE_{trans}^{i,\delta} = \left(\frac{1}{m} \sum_{i=1}^m ||trans(F_i)||^2 \right)^{\frac{1}{2}}$$

$$RPE_{rot}^{i,\delta} = \frac{1}{m} \sum_{i=1}^m \angle (rot(F_i^\delta))$$

где δ — это фиксированный временной интервал.

- Average Translation Error (*средняя ошибка смещения*) [23, 36] и Average Rotation Error (*средняя ошибка поворота*) [23, 36] — показатели разницы в значениях матрицы сдвига и вектора поворота от локальных к глобальным координатам между референсным и оптимизированным значениями. Являются частным случаем Absolute Trajectory Error и Relative Pose Error метрик. Данные две метрики являются наиболее редкоиспользуемыми в соответствующих исследованиях.

Важной деталью такой количественной оценки является то, что все описанные метрики вычисляются с использованием эталонных траекторий определенного набора данных, что приводит к разной точности подсчитанных значений. При визуализации облаков точек со значениями эталонных траекторий также заметны соответствующие дефекты.

В случае отсутствия эталонной траектории лидара, а также в дополнение к референсному подходу, используется безреференсный способ оценивания качества работы SLAM-алгоритма, основанный на визуаль-

ном сравнении положений облаков точек до и после оптимизаций. При таком качественном анализе ключевым становится фиксация и пометка визуальных дефектов, возникающих при наложении облаков точек друг на друга. Данная фиксация помогает отслеживать то, как данные дефекты пропадают или остаются при использовании оптимизаций, на основе чего можно делать выводы о качестве работы SLAM-алгоритма.

2.5. Инструменты для вычисления метрик

Ввиду достаточной сложности и количестве нюансов реализации при подсчете вышеописанных метрик, существует множество инструментов с открытым исходным кодом, позволяющих сделать это максимально просто для пользователя. Основными требованиями, предъявляемыми к подобным библиотекам, являются: возможность максимально подробно вычислить значение метрики, а также визуализировать полученную траекторию; поддерживаемость сообществом. Структурированное сравнение рассматриваемых инструментов для вычисления SLAM-метрик представлено в таблице 3.

Инструмент	Количество звезд	Визуализация	ATE	RPE
evo [28]	3100	+	+	+
kitti-odom-eval [70]	189	+	+	+
evaluate_ate_scale [31]	147	+	+	+
Evaluation_Tools [13]	20	+	+	+
evo_slam [88]	31	+	+	+
localization_analyzer [4]	10	+	+	+

Таблица 3: Инструменты для вычисления SLAM-метрик

Описанные в таблице 3 инструменты не позволяют вычислить значения Average Translation Error и Average Rotation Error метрик, что не является критичным, ввиду недостаточной показательности и редкого использования в тематических статьях. Фаворитом же среди данных инструментов является пакет `evo`. Помимо своей внушительной функциональности, а также большого набора параметров, он активно поддерживается сообществом и автором, что позволяет быстро решать

возникающие вопросы.

2.6. Вывод

Таким образом можно сделать вывод, что воксельные SLAM-алгоритмы в последнее время имеют тенденцию развития (рис. 4) и схожие идеи в реализациях без видимой возможности интеграции между собой, что затрудняет постановку экспериментов с их использованием. Отсутствие возможности расширения или добавления новых модулей в существующие реализации также делает процесс выбора подходящего решения непрозрачным.

Количество наборов данных и метрик для оценки работы SLAM-алгоритмов наоборот является достаточным для проведения экспериментального исследования.

3. Библиотека с воксельными SLAM-алгоритмами

Ключевой идеей работы воксельных SLAM-алгоритмов является оперирование структурированными в виде кубов группами точек. На этапе работы фронтенда происходит распределение точек по вокселям на основе вычисляемого значения планарности с последующим извлечением плоскости. Бэкенд же с помощью плоскостей, извлеченных из облака точек, распределенных по вокселям, оптимизирует положение автономной системы в среде. В данном разделе представлено проектирование и детали разработки библиотеки с компонентами воксельных SLAM-алгоритмов.

3.1. Архитектура библиотеки

Для решения проблемы отсутствия адаптивной конфигурации, расширения, комбинирования существующих воксельных SLAM-алгоритмов с использованием плоскостей была спроектирована соответствующая библиотека, архитектура которой представлена на рисунке 7. Основными требованиями, предъявляемыми к разрабатываемой библиотеке, являются: модульность и адаптивность — для возможности быстрого прототипирования нового решения; повторяемость и воспроизводимость результатов — для того, чтобы результат работы алгоритма можно было повторить не прибегая к кардинальным изменениям в коде. Реализация содержит семь компонентов: `Subdivider`, `Segmenter`, `Backend`, `Pipeline`, `PipelineConfiguration`, `DatasetReader`, `PoseReadWrite`. Рассмотрим их подробнее.

Компонент `Subdivider` содержит реализации условий деления вокселей на октетты. Абстрактный класс, также содержащийся в данном компоненте, содержит всего один абстрактный метод, принимающий и возвращающий точки, за счет чего достигается возможность при реализации сосредоточиться только на сути условия деления, не задумываясь об архитектурных нюансах. Данный компонент содержит следующие

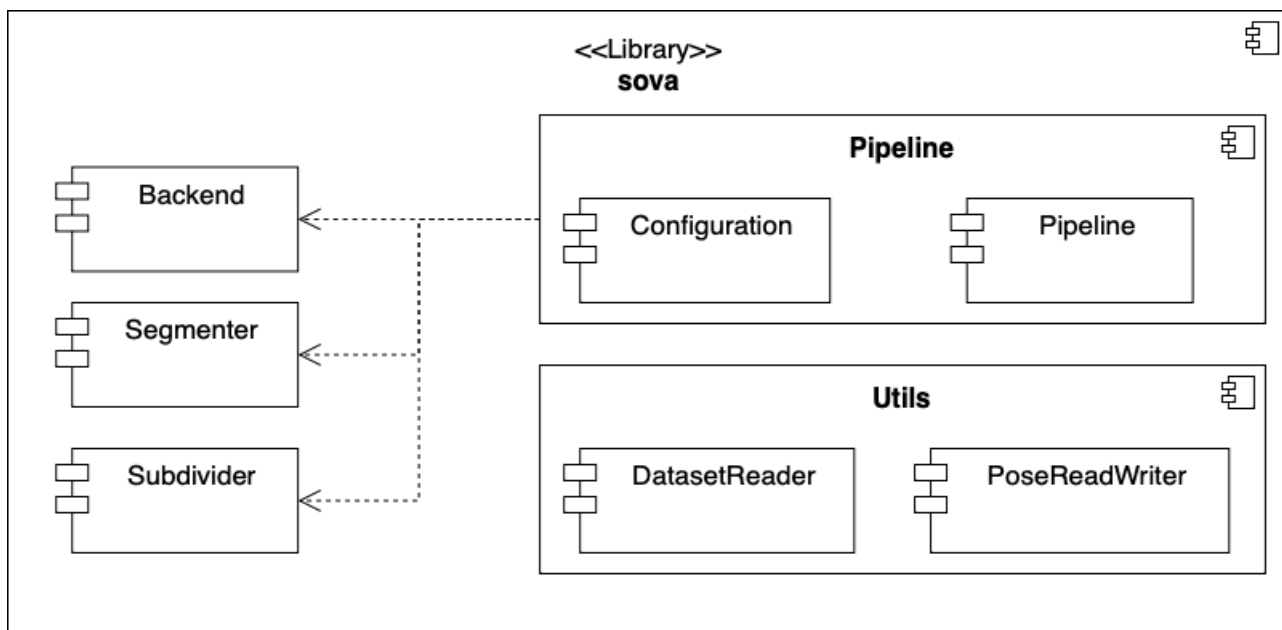


Рис. 7: Архитектура библиотеки

реализации условий деления вокселей: на основе количества точек в нем — **Count**; размера вокселя — **Size**; минимального собственного числа матрицы ковариации точек, содержащихся в вокселе — **EigenValue**.

Компонент **Segmenter** предоставляет алгоритмы по извлечению плоскостей из полученных вокселей. Методы данного компонента следующие: **RANSAC**, выделяющий плоскость согласно идеям, описанным в секции «Обзор»; **CAPE**, выделяющий плоскость на основе соотношения максимального и минимального собственных чисел матрицы ковариации точек; **Identical**, созданный для отладочных целей и возвращающий переданный список точек без изменений.

Компонент **Backend** содержит абстрактный класс, а также различные реализации бэкенд алгоритмов для оптимизации положений лида-ра по выделенным из облаков точек плоскостям. В качестве прикладного решения был выбран пакет **mrob**, так как в нем реализованы все современные типы бэкендов, помимо этого он поддерживается лабораторией мобильной робототехники в Сколтехе.

Компонент **Pipeline** предоставляет реализации воксельных SLAM-алгоритмов с использованием вышеописанных компонентов. **PipelineConfiguration** позволяет производить конфигурирование этих алгоритмов декларативным способом, через **yaml** файлы, за счет

чего можно добиться большей воспроизводимости и повторяемости результатов. Текущая реализация SLAM-алгоритма представлена в виде класса `SequentialPipeline`, обрабатывающим и оптимизирующим облака точек пачками. Схематичный пример работы `SequentialPipeline` представлен на рисунке 8.

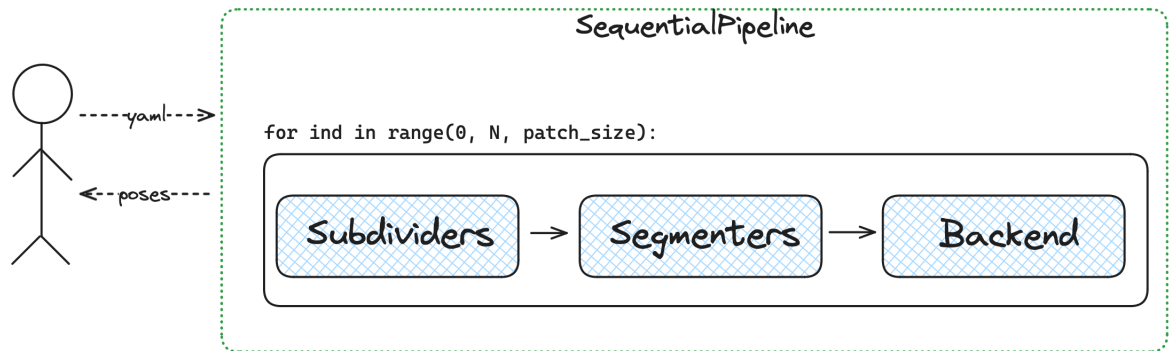


Рис. 8: Пример работы `SequentialPipeline`

Компоненты `DatasetReader` и `PoseReadWriter` являются служебными и предназначены для работы с определенными форматами входных и выходных данных. Методы в компоненте `DatasetReader` позволяют читать облака точек и траектории лидара выбранного набора данных, а модуль `PoseReadWriter` позволяет записывать их оптимизированные значения в указанное место с возможностью последующего чтения.

Помимо описанных компонентов, библиотека также нуждается в реализации структуры данных октодерева, которая бы позволяла эффективно выполнять операции над вокселями с распределенными по ним точками. Этой задачей занимается студент третьего курса направления «Технологии программирования» Киселев Михаил Николаевич.

3.2. Реализация библиотеки

В качестве языка для разработки библиотеки был выбран Python в силу инструментов, позволяющих писать с легкостью сложные конструкции, а также большого количества библиотек для работы с большими объемами данных. Для работы с облаками точек была выбрана библиотека Open3D [79], в силу популярности в мире SLAM-алгоритмов,

а для взаимодействия с большими массивами - фреймворк NumPy [3] в силу большого количества алгоритмов, реализованных в нем. В проекте также написаны юнит-тесты с использованием библиотеки `pytest` [86]. В проекте настроен CI/CD конвейер в GitHub Actions [24], проверяющий стиль кода, запускающий юнит-тесты на платформе Linux (Ubuntu), а также публикующий собранный `pip` пакет на PyPi с использованием библиотеки `poetry` [51]. Список линтеров, используемых в проекте: `black` [85], `isort` [56], `flake8` [55]. Проект распространяется под лицензией Apache 2.0.

4. Экспериментальное исследование

В данном разделе представлено экспериментальное исследование качества и результатов работы SLAM-алгоритмов с использованием плоскостей. Для сравнения полученных результатов используется как визуальная оценка, полученная путем объединения облаков точек разных цветов в одну карту, так и численные показатели метрик, вычисленные с помощью инструмента evo [28].

4.1. Цель и вопросы эксперимента

Целью настоящего эксперимента является сравнение качества работы различных стратегий воксельных SLAM-алгоритмов между собой, а также с другими подходами к решению планарной SLAM-задачи. Для достижения данной цели были поставлены следующие исследовательские вопросы.

- *RQ1: Каково качество оптимизаций положений автономной системы в пространстве при использовании различных стратегий воксельного SLAM-алгоритма?* Данный вопрос нацелен на сравнение результатов работы различных стратегий воксельного SLAM-алгоритма и выявление их недочетов.
- *RQ2: Каково качество работы стратегий воксельного SLAM-алгоритма, по сравнению с другими подходами?* Данный вопрос нацелен на сравнительный анализ разработанных стратегий воксельного SLAM-алгоритма с существующими решениями, выявлению их преимуществ и недостатков.

4.2. Условия эксперимента

Рассмотрим детальнее стенд, на котором производилось экспериментальное исследование.

Наборы данных

Экспериментальное исследование производилось на восьми последовательностях наборов данных Hilti [82] и Newer College Dataset [46], детальное описание которых представлено в таблице 4.

Последовательность	Протяженность (м)	Количество облаков точек	Количество эталонных позиций лидара
Drone Testing Arena (Hilti 2021)	75.6	895	893
Basement1 (Hilti 2021)	59.7	1130	5
Basement4 (Hilti 2021)	105.9	3503	17
Lab (Hilti 2021)	40.0	1357	1338
Construction Site Outdoor 2 (Hilti 2021)	199.2	3991	19
Campus 2 (Hilti 2021)	181.8	3749	19
Short Experiment 0 (Newer College 2020)	165.4	1670	1503
Long Experiment 0 (Newer College 2020)	142.3	1670	1502

Таблица 4: Описание последовательностей, используемых в экспериментах

Отличительной чертой некоторых последовательностей набора данных Hilti, как, например, Basement1, является то, что эталонных позиций лидара в пространстве в количественном соотношении сильно меньше, чем самих облаков точек. Более того, у представленных эталонных значений для указанных последовательностей отсутствует информация о повороте автономной системы в пространстве, что делает невозможным вычисление метрики RPE_{rot} . Тем не менее для вычисления сдвига полученной траектории относительно эталонной предоставленных значений достаточно.

Рассматриваемые алгоритмы

С использованием разработанной Python-библиотеки были получены результаты для пяти различных стратегий, использующих в каче-

стве условия планарности вокселя: его конечный размер в метрах, а также минимальное собственное число матрицы ковариации. В таблице 5 представлено суммарное время, затрачиваемое на вычисление условия планарности вокселей в одном облаке точек в зависимости от используемых параметров. В качестве алгоритма, извлекающего плоскости из полученных вокселей, для всех стратегий использовался метод RANSAC с параметрами: граничное значение, равное 0.01 метра, исходя из точности используемых наборов данных; число опорных точек, равное шести; количество итераций для нахождения наилучшей плоскости, равное 200 согласно работе [33]. В качестве бэкенда был использован класс EigenFactor пакета mrob с методом оптимизации Левенберга-Марквардта и параметром устойчивой оптимизаций QUADRATIC. Количество циклов полной оптимизации, на которых используются предыдущие результаты оптимизированных положений лидара, равно двум.

Стратегия	Начальный размер вокселя (м)	Значение параметра	Вычисление условия (сек.)
Конечный размер вокселя	8	0.5	1.790 ± 0.306
		1	0.821 ± 0.025
		2	0.489 ± 0.013
Минимальное собственное число матрицы ковариации	4	0.5	0.405 ± 0.018
		1	0.018 ± 0.010

Таблица 5: Затраченное время на вычисление условия планарности вокселей с последующим распределением по ним точек. Размер облака точек, на котором производились измерения — 131000; количество измерений — 10.

Для сравнения с существующими аналогами, использующими другой подход к решению SLAM-задачи, были выбраны алгоритмы: FAST_LIO2 [18], BALM [40, 39], LLOL [41], Prime-SLAM [80], так как они являются одними из последних разработанных инструментов, решающих SLAM-задачу. Из описанных алгоритмов удалось запустить лишь FAST_LIO2 и Prime-SLAM: с помощью алгоритма BALM не удалось добиться каких-либо оптимизаций, ввиду слишком большого начального значения вычисляемой ошибки внутри самого алгоритма, а алгоритм llo1 более не поддерживается автором и зависит от неизвестных версий устаревших библиотек, поэтому попытка сборки данного проекта

была неуспешной.

Характеристики системы

Для запуска всех SLAM-алгоритмов использовалась машина: AMD Ryzen 7 4800H @ 2.9 ГГц (отключенный Turbo Boost), 24 Gb RAM, NVIDIA GeForce RTX 3050 (mobile), Ubuntu 22.04.

4.3. Исследование SLAM-алгоритмов

В данном разделе представлено качественное и количественное сравнение результатов работы рассматриваемых SLAM-алгоритмов.

Экспериментальное исследование стратегий воксельных SLAM-алгоритмов

В качестве первоначальных позиций лидара в пространстве для разработанных стратегий воксельного SLAM-алгоритма были взяты значения, полученные в результате работы двух алгоритмов: FAST_LIO2 и Prime-SLAM. При использовании минимального собственного числа матрицы ковариации в качестве условия планарности вокселя не удалось добиться значительных оптимизаций на длинных последовательностях, ввиду чего данный параметр упускается из рассмотрения. На рисунке 9 представлен частный пример визуального сравнения качества работы рассматриваемых стратегий воксельного SLAM-алгоритма на примере одного из дефектов, возникших на последовательности Drone. Данный дефект возникает при объединении части облаков точек последовательности и представляет собой стык двух стен. Все рассматриваемые стратегии хорошо справляются с устранением данного дефекта, хоть и заметно, что с ростом значения параметров точность работы алгоритма уменьшается. Лучший результат из представленных достигается при использовании в качестве условия планарности вокселя значения, равного 0.5, но, как можно видеть из таблицы 5, оно не является самым производительным, уступая всем остальным стратегиям.

Несмотря на рассмотренный частный пример, использование значения 0.5 метра в качестве конечного размера вокселя не является универсальным параметром и применимо для небольших последовательностей и малых помещений, на длительных же последовательностях использование данного параметра приведет к избыточному количеству ориентиров, в результате чего бэкенду становится трудно построить полноценную карту.

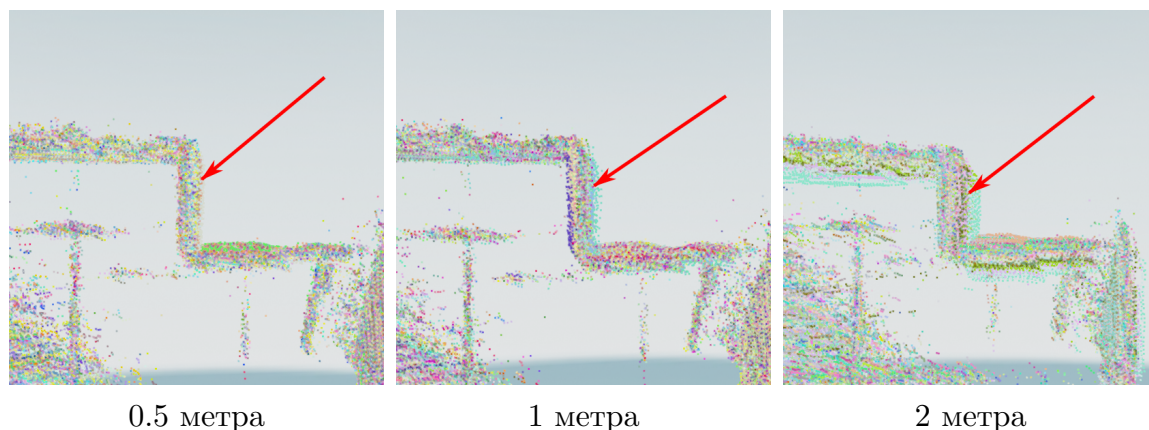


Рис. 9: Пример результата работы стратегий воксельных SLAM-алгоритмов при различных параметрах для последовательности Drone

Результаты для всех рассматриваемых последовательностей, а также соответствующие параметры, с которыми запускались алгоритмы, представлены в таблице A.1 (Приложение A). В данной таблице жирным шрифтом выделены значения метрик лучших стратегий, имеющих меньшее количество визуальных дефектов на той или иной последовательности. Как было ранее указано в секции «Метрики точности работы SLAM-алгоритмов», их значения посчитаны с точностью до эталонных траекторий, имеющих погрешность и значительные визуальные дефекты, как представлено на рисунке 10. Конечный размер вокселя равный 0.5 метра использовался лишь для одной последовательности — Drone Testing Arena, ввиду специфичности последовательности и небольшого размера комнаты, в которой летал дрон, на других последовательностях использование данного параметра не позволило добиться значительных оптимизаций.

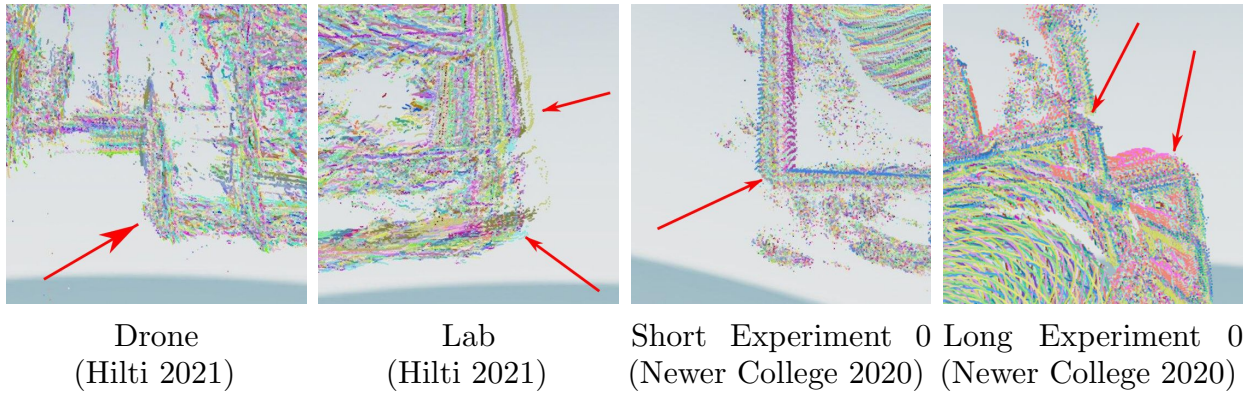


Рис. 10: Примеры визуальных дефектов в случае использования эталонных позиций и поворотов лидара в пространстве

Исходя из полученных результатов можно сделать вывод, что увеличение конечного размера вокселя с единицы до двух влечет за собой рост метрики ATE и увеличение количества визуальных дефектов, а на некоторых приводит к сильному ухудшению результата. Такое поведение обусловлено тем, что фронтенд извлекает слишком мало ориентиров для качественной оптимизации, в результате чего итоговая траектория ухудшается. Корреляция между количественными результатами при использовании различных размеров одного участка оптимизаций особенно заметна на последовательностях Basement1, Basement4 и Construction Site 2, что также выражается во многих визуальных дефектах. Такое качественное и количественное ухудшение результата связано с тем, что бэкенду передается слишком много ориентиров, из-за чего ему становится трудно построить полноценную карту.

Экспериментальное сравнение с существующими аналогами

В таблице 6 представлены значения метрик точности работы рассматриваемых SLAM-алгоритмов. Из данной таблицы видно, что благодаря использованию разработанной библиотеки получается значительно уменьшить значение RPE_{rot} , отвечающей за относительный поворот облаков точек. Значения метрик ATE и RPE_{trans} практически не изменяются, либо изменяются в пределах погрешности эталонных значе-

Последовательность	ATE (м)			RPE (м)						RPE (deg)		
	FAST_LIO2			FAST_LIO2 + Voxel-SLAM			Prime-SLAM			Prime-SLAM + Voxel-SLAM		
Drone Testing Arena (Hilti 2021)	0.181	0.008	0.55	0.191	0.027	0.45	0.452	0.019	0.33	0.435	0.045	0.42
Basement1 (Hilti 2021)	0.158	0.138	-	0.159	0.136	-	0.229	0.062	-	0.230	0.064	-
Basement4 (Hilti 2021)	0.037	0.010	-	0.039	0.020	-	0.289	0.105	-	0.290	0.110	-
Lab (Hilti 2021)	0.029	0.009	0.73	0.059	0.015	0.96	0.153	0.013	0.94	0.137	0.023	0.94
Construction Site Outdoor 2 (Hilti 2021)	0.091	0.067	-	0.083	0.060	-	0.960	0.434	-	0.957	0.448	-
Campus 2 (Hilti 2021)	0.085	0.026	-	0.087	0.031	-	3.690	0.269	-	3.699	0.267	-
Short Experiment 0 (Newer College 2020)	0.51	0.09	1.38	0.50	0.06	0.73	5.90	0.05	1.28	5.70	0.13	0.80
Long Experiment 0 (Newer College 2020)	0.45	0.08	2.11	0.45	0.06	1.17	5.94	0.06	2.09	5.93	0.23	1.11

Таблица 6: Результаты запуска SLAM-алгоритмов, посчитанных с использованием эталонных значений. Жирным шрифтом выделены наименьшие значения RPE_{rot} из рассматриваемых пар алгоритмов

ний, ввиду чего можно сделать вывод, что воксельный подход сохраняет заданную ему траекторию. Описанная динамика в результатах также подтверждается рисунком 11, из которого видно, как визуальные дефекты, возникающие при использовании алгоритмов FAST_LIO2 и Prime-SLAM, корректируются благодаря воксельному подходу. Пример сохранения траектории после применения воксельного подхода продемонстрирован на рисунке 12 (Приложение А).

Исходя из полученных результатов можно сделать вывод, что использование связки алгоритмов FAST_LIO2 и Voxel-SLAM дает наилучший качественный и количественный результат, в силу использования облаков точек в качестве данных на протяжении всей работы. FAST_LIO2 задает траекторию близкую к эталонной, после чего Voxel-SLAM способен устранить возникшие визуальные дефекты и ошибки поворота автономной системы в пространстве. Алгоритм Prime-SLAM при извлечении ориентиров полагается на стереоснимки, ввиду чего ему бывает порой трудно построить карту из-за плохой освещенности или недостатка ориентиров, как это случилось, например, в конце последовательности Basement1. Тем не менее использование воксельного подхода на траектории, полученной от Prime-SLAM, позволяет добиться

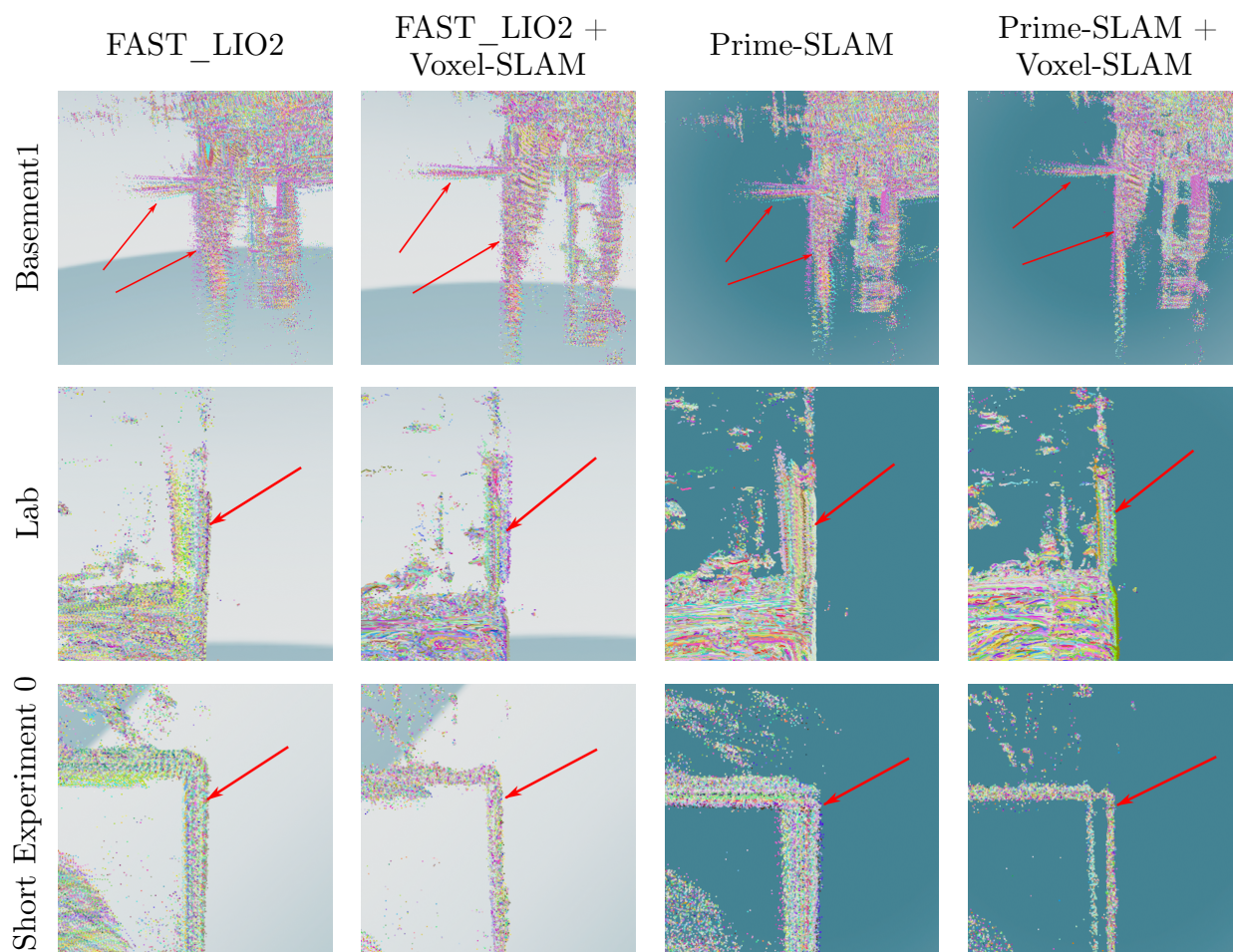


Рис. 11: Пример коррекции визуальных дефектов после использования воксельного подхода

более качественных визуальных положений облаков точек в пространстве без ущерба вычисляемым метрикам.

4.4. Вывод

Подведем итоги результатов экспериментального сравнения различных подходов к решению SLAM-алгоритмов, ответив на поставленные исследовательские вопросы.

RQ1: Каково качество оптимизаций положений автономной системы в пространстве при использовании различных стратегий воксельного SLAM-алгоритма? Исходя из результатов, полученных в ходе экспериментов со стратегиями воксельного SLAM-алгоритма, можно сделать вывод, что их использование улучшает глобальный поворот лида-

ра в пространстве и позволяет избавиться от первоначальных визуальных дефектов, не добавляя новых. Возможность конфигурировать параметры стратегий воксельного SLAM-алгоритма позволяет количественно проследить динамику изменения тех или иных значений метрик, а также визуально оценить полученные оптимизации.

RQ2: Каково качество работы стратегий воксельного SLAM-алгоритма, по сравнению с другими подходами? Исходя из результатов, полученных применением алгоритмов FAST_LIO2, Prime-SLAM и Voxel-SLAM, можно сделать вывод, что воксельные SLAM-алгоритмы позволяют значительно улучшить оценку поворота автономной системы в пространстве на протяжении всех последовательностей, сохраняя заданную траекторию, вне зависимости от использования первоначального SLAM-алгоритма.

5. Заключение

В результате работы были решены следующие задачи.

1. Проведен обзор существующих подходов к извлечению плоскостей из облаков точек, решений планарной SLAM-задачи, метрик, оценивающих результаты их работы, наборов данных, на которых проводятся эксперименты.
2. Спроектирована и разработана модульная Python-библиотека³, позволяющая разрабатывать и поддерживать различные компоненты воксельных SLAM-алгоритмов с использованием плоскостей для запуска различных стратегий.
3. Проведено экспериментальное исследование различных стратегий воксельных SLAM-алгоритмов с использованием плоскостей.
4. Проведено экспериментальное сравнение разработанных стратегий воксельных SLAM-алгоритмов с существующими решениями SLAM-задачи.

³<https://github.com/prime-slam/sova> (дата обращения: 01.05.2024).

Список литературы

- [1] The 3d hough transform for plane detection in point clouds: A review and a new accumulator design / Dorit Borrmann, Jan Elseberg, Kai Lingemann, Andreas Nüchter // 3D Research. — 2011. — Vol. 2, no. 2. — P. 1–13.
- [2] Aijazi Ahmad Kamal, Checchin Paul, Trassoudaine Laurent. Segmentation based classification of 3D urban point clouds: A super-voxel based approach with evaluation // Remote Sensing. — 2013. — Vol. 5, no. 4. — P. 1624–1650.
- [3] Array programming with NumPy / Charles R. Harris, K. Jarrod Millman, Stéfan J. van der Walt et al. // *Nature*. — 2020. — September. — Vol. 585, no. 7825. — P. 357–362. — URL: <https://doi.org/10.1038/s41586-020-2649-2> (дата обращения: 22.05.2024).
- [4] Autonomy NeBula. Localization Analyzer // Repository. — URL: https://github.com/NeBula-Autonomy/localization_analyzer (дата обращения: 22.05.2024).
- [5] Ballard Dana H. Generalizing the Hough transform to detect arbitrary shapes // Pattern recognition. — 1981. — Vol. 13, no. 2. — P. 111–122.
- [6] Besl Paul J., Jain Ramesh C. Segmentation through variable-order surface fitting // IEEE Transactions on pattern analysis and machine intelligence. — 1988. — Vol. 10, no. 2. — P. 167–192.
- [7] Cai Yixi, Xu Wei, Zhang Fu. ikd-tree: An incremental kd tree for robotic applications // arXiv preprint arXiv:2102.10808. — 2021.
- [8] Chehata Nesrine, David Nicolas, Bretar Frédéric. LIDAR data classification using hierarchical K-means clustering // ISPRS Congress Beijing 2008 / Citeseer. — Vol. 37. — 2008. — P. 325–330.

- [9] Chen Jianhong, Wang Hongwei, Yang Shan. Tightly Coupled LiDAR-Inertial Odometry and Mapping for Underground Environments // Sensors. — 2023. — Vol. 23, no. 15. — P. 6834.
- [10] Chum Ondřej, Matas Jiří. Optimal randomized RANSAC // IEEE Transactions on Pattern Analysis and Machine Intelligence. — 2008. — Vol. 30, no. 8. — P. 1472–1482.
- [11] Chum Ondřej, Matas Jiří, Kittler Josef. Locally optimized RANSAC // Pattern Recognition: 25th DAGM Symposium, Magdeburg, Germany, September 10-12, 2003. Proceedings 25 / Springer. — 2003. — P. 236–243.
- [12] Clustering-based plane refitting of non-planar patches for voxel-based 3D point cloud segmentation using k-means clustering / Ali Sağlam, Hasan Bilgehan Makineci, Ömer Kaan Baykan, Nurdan Akhan Baykan // Traitement du Signal. — 2020.
- [13] Dai Weichen. Estimation tools for visual odometry or SLAM // Repository. — URL: https://github.com/weichnn/Evaluation_Tools (дата обращения: 22.05.2024).
- [14] Deschaud Jean-Emmanuel, Goulette François. A fast and accurate plane detection algorithm for large noisy point clouds using filtered normals and voxel growing // 3DPVT. — 2010.
- [15] DurLAR: A High-fidelity 128-channel LiDAR Dataset with Panoramic Ambient and Reflectivity Imagery for Multi-modal Autonomous Driving Applications / L. Li, K.N. Ismail, H.P.H. Shum, T.P. Breckon // Proc. Int. Conf. on 3D Vision. — IEEE, 2021. — December.
- [16] Efficient and probabilistic adaptive voxel mapping for accurate online lidar odometry / Chongjian Yuan, Wei Xu, Xiyuan Liu et al. // IEEE Robotics and Automation Letters. — 2022. — Vol. 7, no. 3. — P. 8518–8525.

- [17] Eustice Ryan M, Singh Hanumant, Leonard John J. Exactly sparse delayed-state filters // Proceedings of the 2005 IEEE International Conference on Robotics and Automation / IEEE. — 2005. — P. 2417–2424.
- [18] Fast-lio2: Fast direct lidar-inertial odometry / Wei Xu, Yixi Cai, Dongjiao He et al. // IEEE Transactions on Robotics. — 2022. — Vol. 38, no. 4. — P. 2053–2073.
- [19] Fast planar surface 3D SLAM using LIDAR / Kruno Lenac, Andrej Kitanov, Robert Cupec, Ivan Petrović // Robotics and Autonomous Systems. — 2017. — Vol. 92. — P. 197–220.
- [20] Ferrer Gonzalo, Iarosh Dmitrii, Kornilova Anastasiia. Eigen-Factors an Alternating Optimization for Back-end Plane SLAM of 3D Point Clouds // arXiv preprint arXiv:2304.01055. — 2023.
- [21] Fischler Martin A, Bolles Robert C. Random sample consensus: a paradigm for model fitting with applications to image analysis and automated cartography // Communications of the ACM. — 1981. — Vol. 24, no. 6. — P. 381–395.
- [22] Fuhrmann Simon, Langguth Fabian, Goesele Michael. Mve-a multi-view reconstruction environment. // GCH. — 2014. — Vol. 3. — P. 4.
- [23] Geiger Andreas, Lenz Philip, Urtasun Raquel. Are we ready for Autonomous Driving? The KITTI Vision Benchmark Suite // Conference on Computer Vision and Pattern Recognition (CVPR). — 2012.
- [24] GitHub. GitHubActions // Site. — URL: <https://github.com/features/actions> (дата обращения: 22.05.2024).
- [25] Golovinskiy Aleksey, Funkhouser Thomas. Min-cut based segmentation of point clouds // 2009 IEEE 12th International Conference on Computer Vision Workshops, ICCV Workshops / IEEE. — 2009. — P. 39–46.

- [26] Gorte Ben. Segmentation of TIN-structured surface models // International Archives of Photogrammetry Remote Sensing and Spatial Information Sciences. — 2002. — Vol. 34, no. 4. — P. 465–469.
- [27] Grant W Shane, Voorhies Randolph C, Itti Laurent. Finding planes in LiDAR point clouds for real-time registration // 2013 IEEE/RSJ International Conference on Intelligent Robots and Systems / IEEE. — 2013. — P. 4347–4354.
- [28] Grupp Michael. evo: Python package for the evaluation of odometry and SLAM. — 2017. — URL: <https://github.com/MichaelGrupp/evo> (дата обращения: 22.05.2024).
- [29] Guan Yanran (Isaac). Hough Transform for Plane Detection // Site. — URL: <https://isaacguan.github.io/2018/02/28/Hough-Transform-for-Plane-Detection/> (дата обращения: 22.05.2024).
- [30] Hofmann Alexandra D, Maas Hans-Gerd, Streilein André. Derivation of roof types by cluster analysis in parameter spaces of airborne laser-scanner point clouds // IAPRS International Archives of Photogrammetry and Remote Sensing and Spatial Information Sciences. — 2003. — Vol. 34, no. Part 3. — P. W13.
- [31] Horn Berthold KP. Closed-form solution of absolute orientation using unit quaternions // Josa a. — 1987. — Vol. 4, no. 4. — P. 629–642.
- [32] Huang Ming, Wei Pengcheng, Liu Xianglei. An efficient encoding voxel-based segmentation (EVBS) algorithm based on fast adjacent voxel search for point cloud plane segmentation // Remote Sensing. — 2019. — Vol. 11, no. 23. — P. 2727.
- [33] Improving RANSAC for fast landmark recognition / Pablo Márquez-Neila, Jacobo García Miró, José M Buenaposada, Luis Baumela // 2008 IEEE Computer Society Conference on Computer Vision and Pattern Recognition Workshops / IEEE. — 2008. — P. 1–8.

- [34] Khaloo Ali, Lattanzi David. Robust normal estimation and region growing segmentation of infrastructure 3D point cloud models // Advanced Engineering Informatics. — 2017. — Vol. 34. — P. 1–16.
- [35] Kiryati Nahum, Eldar Yuval, Bruckstein Alfred M. A probabilistic Hough transform // Pattern recognition. — 1991. — Vol. 24, no. 4. — P. 303–316.
- [36] LIO-GVM: an Accurate, Tightly-Coupled Lidar-Inertial Odometry with Gaussian Voxel Map / Xingyu Ji, Shenghai Yuan, Pengyu Yin, Lihua Xie // arXiv preprint arXiv:2306.17436. — 2023.
- [37] Large-scale LiDAR SLAM with factor graph optimization on high-level geometric features / Krzysztof Ćwian, Michał R Nowicki, Jan Wietrzykowski, Piotr Skrzypczyński // Sensors. — 2021. — Vol. 21, no. 10. — P. 3445.
- [38] Li Minglei, Sun Changming. Refinement of LiDAR point clouds using a super voxel based approach // ISPRS Journal of Photogrammetry and Remote Sensing. — 2018. — Vol. 143. — P. 213–221.
- [39] Liu Zheng, Liu Xiyuan, Zhang Fu. Efficient and consistent bundle adjustment on Lidar point clouds // IEEE Transactions on Robotics. — 2023.
- [40] Liu Zheng, Zhang Fu. Balm: Bundle adjustment for lidar mapping // IEEE Robotics and Automation Letters. — 2021. — Vol. 6, no. 2. — P. 3184–3191.
- [41] LloL: Low-latency odometry for spinning lidars / Chao Qu, Shreyas S Shivakumar, Wenxin Liu, Camillo J Taylor // 2022 International Conference on Robotics and Automation (ICRA) / IEEE. — 2022. — P. 4149–4155.
- [42] Matas Jiri, Chum Ondrej. Randomized RANSAC with sequential probability ratio test // Tenth IEEE International Conference on Computer Vision (ICCV'05) Volume 1 / IEEE. — Vol. 2. — 2005. — P. 1727–1732.

- [43] Matasyx J, Kittlery C Galambosyand J. Progressive probabilistic hough transform // Proceedings of the British Machine Vision Conference, Southampton, UK. — 1998. — P. 14–17.
- [44] Measuring robustness of Visual SLAM / David Prokhorov, Dmitry Zhukov, Olga Barinova et al. // 2019 16th International Conference on Machine Vision Applications (MVA) / IEEE. — 2019. — P. 1–6.
- [45] Mendes Ellison, Koch Pierrick, Lacroix Simon. ICP-based pose-graph SLAM // 2016 IEEE International Symposium on Safety, Security, and Rescue Robotics (SSRR) / IEEE. — 2016. — P. 195–200.
- [46] Zhang Lintong, Camurri Marco, Wisth David, Fallon Maurice. Multi-Camera LiDAR Inertial Extension to the Newer College Dataset. — 2021. — 2112.08854.
- [47] Octree-based region growing for point cloud segmentation / Anh-Vu Vo, Linh Truong-Hong, Debra F Laefer, Michela Bertolotto // ISPRS Journal of Photogrammetry and Remote Sensing. — 2015. — Vol. 104. — P. 88–100.
- [48] On bundle adjustment for multiview point cloud registration / Huaiyang Huang, Yuxiang Sun, Jin Wu et al. // IEEE Robotics and Automation Letters. — 2021. — Vol. 6, no. 4. — P. 8269–8276.
- [49] Panoptic nuScenes: A Large-Scale Benchmark for LiDAR Panoptic Segmentation and Tracking / Whye Kit Fong, Rohit Mohan, Juana Valeria Hurtado et al. // arXiv preprint arXiv:2109.03805. — 2021.
- [50] Plane segmentation and fitting method of point clouds based on improved density clustering algorithm for laser radar / Xiaobin Xu, Minzhou Luo, Zhiying Tan et al. // Infrared Physics & Technology. — 2019. — Vol. 96. — P. 133–140.
- [51] Poetry. Python packaging and dependency management made easy // Repository. — URL: <https://github.com/python-poetry/poetry> (дата обращения: 22.05.2024).

- [52] Point-plane SLAM for hand-held 3D sensors / Yuichi Taguchi, Yong-Dian Jian, Srikumar Ramalingam, Chen Feng // 2013 IEEE international conference on robotics and automation / IEEE. — 2013. — P. 5182–5189.
- [53] Poux Florent, Billen Roland. Voxel-based 3D point cloud semantic segmentation: Unsupervised geometric and relationship featuring vs deep learning methods // ISPRS International Journal of Geo-Information. — 2019. — Vol. 8, no. 5. — P. 213.
- [54] Proença Pedro F, Gao Yang. Fast cylinder and plane extraction from depth cameras for visual odometry // 2018 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS) / IEEE. — 2018. — P. 6813–6820.
- [55] PyCQA. flake8 // Site. — URL: <https://flake8.pycqa.org/en/latest/> (дата обращения: 22.05.2024).
- [56] PyCQA. isort // Site. — URL: <https://pycqa.github.io/isort/> (дата обращения: 22.05.2024).
- [57] Rabbani Tahir, Van Den Heuvel Frank. Efficient hough transform for automatic detection of cylinders in point clouds // Isprs Wg Iii/3, Iii/4. — 2005. — Vol. 3. — P. 60–65.
- [58] Rabbani Tahir, Van Den Heuvel Frank, Vosselmann George. Segmentation of point clouds using smoothness constraint // International archives of photogrammetry, remote sensing and spatial information sciences. — 2006. — Vol. 36, no. 5. — P. 248–253.
- [59] Ryde Julian, Dhiman Vikas, Platt Robert. Voxel planes: Rapid visualization and meshification of point cloud ensembles // 2013 IEEE/RSJ International Conference on Intelligent Robots and Systems / IEEE. — 2013. — P. 3731–3737.

- [60] Salti Samuele, Tombari Federico, Di Stefano Luigi. SHOT: Unique signatures of histograms for surface and texture description // Computer Vision and Image Understanding. — 2014. — Vol. 125. — P. 251–264.
- [61] Scalability in Perception for Autonomous Driving: Waymo Open Dataset / Pei Sun, Henrik Kretzschmar, Xerxes Dotiwalla et al. // CoRR. — 2019. — Vol. abs/1912.04838. — arXiv : [1912.04838](https://arxiv.org/abs/1912.04838).
- [62] Schnabel Ruwen, Wahl Roland, Klein Reinhard. Efficient RANSAC for point-cloud shape detection // Computer graphics forum / Wiley Online Library. — Vol. 26. — 2007. — P. 214–226.
- [63] Spherical parameter detection based on hierarchical Hough transform / MY Cao, CH Ye, Olaf Doessel, C Liu // Pattern recognition letters. — 2006. — Vol. 27, no. 9. — P. 980–986.
- [64] Stephens Richard S. Probabilistic approach to the Hough transform // Image and vision computing. — 1991. — Vol. 9, no. 1. — P. 66–71.
- [65] Su Yun-Ting, Bethel James, Hu Shuowen. Octree-based segmentation for terrestrial LiDAR point cloud data in industrial applications // ISPRS Journal of Photogrammetry and Remote Sensing. — 2016. — Vol. 113. — P. 59–74.
- [66] Tóvári Daniel, Pfeifer Norbert. Segmentation based robust interpolation-a new approach to laser data filtering // International Archives of Photogrammetry, Remote Sensing and Spatial Information Sciences. — 2005. — Vol. 36, no. 3/19. — P. 79–84.
- [67] Trevor Alexander J. B., Rogers John G., Christensen Henrik I. [Planar surface SLAM with 3D and 2D sensors](#) // 2012 IEEE International Conference on Robotics and Automation. — 2012. — P. 3041–3048.
- [68] Vieira Miguel, Shimada Kenji. Surface mesh segmentation and smooth surface extraction through region growing // Computer aided geometric design. — 2005. — Vol. 22, no. 8. — P. 771–792.

- [69] Vins-mkf: A tightly-coupled multi-keyframe visual-inertial odometry for accurate and robust state estimation / Chaofan Zhang, Yong Liu, Fan Wang et al. // *Sensors*. — 2018. — Vol. 18, no. 11. — P. 4036.
- [70] Visual Odometry Revisited: What Should Be Learnt? / Huangying Zhan, Chamara Saroj Weerasekera, Jiawang Bian, Ian Reid // *arXiv preprint arXiv:1909.09803*. — 2019.
- [71] VoxelMap++: Mergeable Voxel Mapping Method for Online LiDAR (-inertial) Odometry / Yifei Yuan, Chang Wu, Yuan You et al. // *arXiv preprint arXiv:2308.02799*. — 2023.
- [72] Xu Lei, Oja Erkki. Randomized Hough transform (RHT): basic mechanisms, algorithms, and computational complexities // *CVGIP: Image understanding*. — 1993. — Vol. 57, no. 2. — P. 131–154.
- [73] Xu Wei, Zhang Fu. Fast-lio: A fast, robust lidar-inertial odometry package by tightly-coupled iterated kalman filter // *IEEE Robotics and Automation Letters*. — 2021. — Vol. 6, no. 2. — P. 3317–3324.
- [74] Zhang He, Ye Cang. An indoor wayfinding system based on geometric features aided graph SLAM for the visually impaired // *IEEE Transactions on Neural Systems and Rehabilitation Engineering*. — 2017. — Vol. 25, no. 9. — P. 1592–1604.
- [75] Zhang Juyong, Yao Yuxin, Deng Bailin. Fast and robust iterative closest point // *IEEE Transactions on Pattern Analysis and Machine Intelligence*. — 2021. — Vol. 44, no. 7. — P. 3450–3466.
- [76] Zhang Zhengyou. Iterative closest point (ICP) // *Computer vision: a reference guide*. — Springer, 2021. — P. 718–720.
- [77] Zhou Lipu, Koppel Daniel, Kaess Michael. LiDAR SLAM with plane adjustment for indoor environment // *IEEE Robotics and Automation Letters*. — 2021. — Vol. 6, no. 4. — P. 7073–7080.

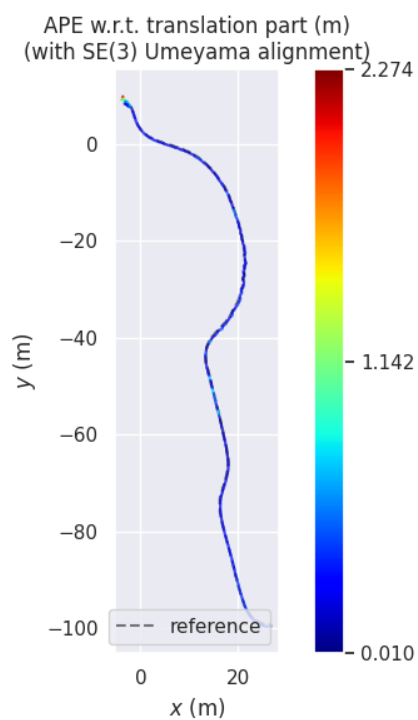
- [78] Zhou Lipu, Wang Shengze, Kaess Michael. π -LSAM: LiDAR smoothing and mapping with planes // 2021 IEEE International Conference on Robotics and Automation (ICRA) / IEEE. — 2021. — P. 5751–5757.
- [79] Zhou Qian-Yi, Park Jaesik, Koltun Vladlen. Open3D: A Modern Library for 3D Data Processing // CoRR. — 2018. — Vol. abs/1801.09847. — arXiv : [1801.09847](https://arxiv.org/abs/1801.09847).
- [80] at Skoltech Mobile Robotics Lab. SLAM system using various types of landmarks. // Repository. — URL: <https://github.com/prime-slam/prime-slam> (дата обращения: 22.05.2024).
- [81] at Skoltech Mobile Robotics Lab. mrob: Cpp-Python library for state estimation in 2D and 3D by solving graph optimization // Repository. — URL: <https://github.com/prime-slam/mrob> (дата обращения: 22.05.2024).
- [82] The hilti slam challenge dataset / Michael Helmberger, Kristian Morin, Beda Berner et al. // IEEE Robotics and Automation Letters. — 2022. — Vol. 7, no. 3. — P. 7518–7525.
- [83] A new segmentation method for point cloud data / H Woo, E Kang, Semyung Wang, Kwan H Lee // International Journal of Machine Tools and Manufacture. — 2002. — Vol. 42, no. 2. — P. 167–178.
- [84] An online multi-robot SLAM system for 3D LiDARs / Renaud Dubé, Abel Gawel, Hannes Sommer et al. // 2017 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS) / IEEE. — 2017. — P. 1004–1011.
- [85] pcf. Black Formatter // Repository. — URL: <https://github.com/psf/black> (дата обращения: 22.05.2024).
- [86] Krekel Holger, Oliveira Bruno, Pfannschmidt Ronny et al. pytest х.у. — 2004. — URL: <https://github.com/pytest-dev/pytest> (дата обращения: 22.05.2024).

- [87] A voxel-based filtering algorithm for mobile LiDAR data / Hejuan Qin, G Guan, Yongtao Yu, Liang Zhong // The International Archives of the Photogrammetry, Remote Sensing and Spatial Information Sciences. — 2018. — Vol. 42. — P. 1433–1438.
- [88] 李洋洋. SLAM evolution tool // Repository. — URL: https://github.com/lyy-ai/evo_slam (дата обращения: 22.05.2024).

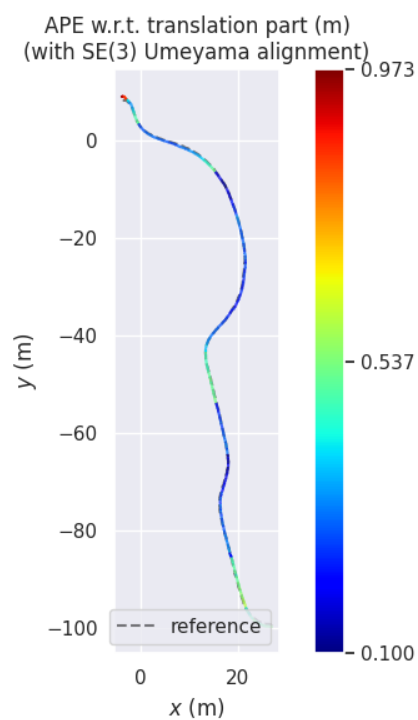
Приложение А

Последовательность	Конфигурация		ATE (м)		RPE (м)		RPE (deg)	
	Размер одного участка оптимизации	Размер вокселя	FAST_LIO2 + Voxel-SLAM		Prime-SLAM + Voxel-SLAM			
Drone Testing Arena (Hilti 2021)	50	0.5	0.191	0.027	0.45	0.460	0.044	0.39
	50	1	0.200	0.032	0.44	0.467	0.050	0.42
	50	2	0.483	0.179	0.48	0.525	0.084	0.43
	100	1	0.211	0.032	0.41	0.434	0.045	0.42
	100	2	0.472	0.153	0.44	0.492	0.064	0.43
Basement1 (Hilti 2021)	50	1	0.160	0.135	-	0.230	0.064	-
	50	2	0.294	0.342	-	0.607	0.479	-
	100	1	0.159	0.136	-	0.234	0.080	-
	100	2	0.453	0.482	-	0.538	0.275	-
Basement4 (Hilti 2021)	50	1	0.039	0.020	-	0.290	0.110	-
	50	2	0.046	0.028	-	0.295	0.108	-
	100	1	0.048	0.049	-	0.293	0.140	-
	100	2	11.103	14.232	-	0.297	0.160	-
Lab (Hilti 2021)	50	1	0.059	0.015	0.96	0.149	0.030	0.93
	50	2	0.140	0.072	1.27	0.194	0.079	1.32
	100	1	0.052	0.014	0.95	0.137	0.023	0.94
	100	2	0.192	0.077	1.28	0.330	0.099	1.59
Construction Site Outdoor 2 (Hilti 2021)	50	1	0.083	0.060	-	0.962	0.430	-
	50	2	0.261	0.263	-	0.966	0.431	-
	100	1	1.895	2.664	-	0.957	0.448	-
	100	2	4.252	5.704	-	7.205	7.992	-
Campus 2 (Hilti 2021)	50	1	0.087	0.031	-	3.693	0.276	-
	50	2	0.089	0.037	-	3.691	0.278	-
	100	1	0.095	0.047	-	3.699	0.267	-
	100	2	0.137	0.122	-	3.726	0.386	-
Short Experiment 0 (Newer College 2020)	50	1	0.50	0.7	0.77	6.08	0.34	0.77
	50	2	0.51	0.08	0.76	5.77	0.13	0.77
	100	1	0.50	0.06	0.73	5.70	0.13	0.80
	100	2	0.51	0.07	0.73	5.65	0.13	0.80
Long Experiment 0 (Newer College 2020)	50	1	0.44	0.07	1.24	5.94	0.18	1.10
	50	2	0.45	0.07	1.25	5.94	0.18	1.10
	100	1	0.46	0.06	1.17	5.92	0.23	1.11
	100	2	0.46	0.07	1.17	5.92	0.23	1.11

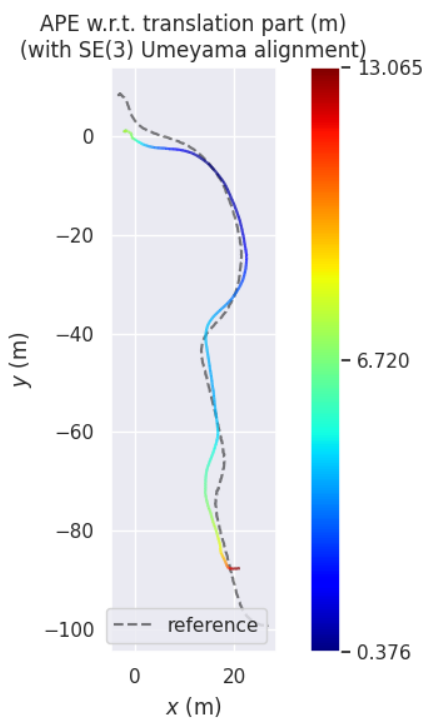
Таблица А.1: Результаты запуска рассматриваемых стратегий воксельного SLAM-алгоритма, посчитанных с использованием эталонных значений



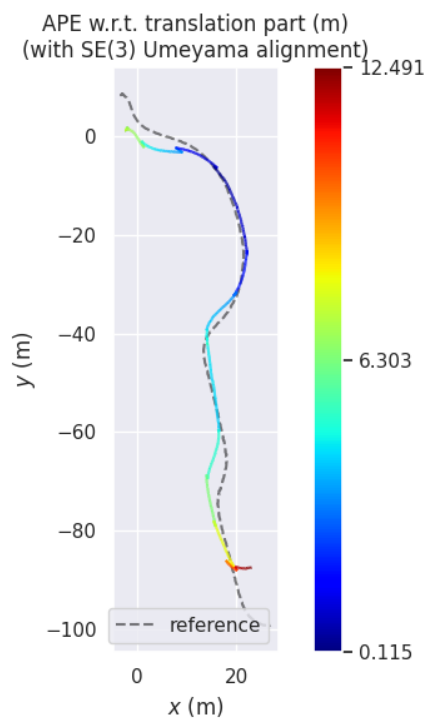
FAST_LIO2



FAST_LIO2 + Voxel-SLAM



Prime-SLAM



Prime-SLAM + Voxel-SLAM

Рис. 12: Пример траекторий последовательности Short Experiment 0, полученных применением рассматриваемых SLAM-алгоритмов