Санкт-Петербургский государственный университет

Салью Артур Кристофович

Выпускная квалификационная работа

Разработка библиотеки извлечения плоскостей из RGB-D снимков

Уровень образования: бакалавриат

Направление *09.03.04 «Программная инженерия»*

Основная образовательная программа *СВ.5080.2019 «Программная инженерия»*

Научный руководитель:
доцент кафедры СП, к.т.н. Ю. В. Литвинов

Рецензент:
научный сотрудник, Мюнхенский технический университет, к.ф.-м.н. А. В. Артемов

Санкт-Петербург
2023

Saint Petersburg State University

Arthur Saliou

Bachelor's Thesis

Development of a library for plane segmentation of RGB-D images

Education level: bachelor

Speciality *09.03.04 "Software Engineering"*

Programme *CB.5080.2019 "Software Engineering"*

Scientific supervisor:
C.Sc., System Programming chair docent Y. V. Litvinov

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

Saint Petersburg
2023

Оглавление

Введение

Задача определения положения автономной системы и картирования окружающей её среды является одной из базовых для успешного оперирования этой системы. Активно развивающимся подходом для решения этой задачи является класс алгоритмов одновременной локализации и построения карты (SLAM — *Simultaneous localization and mapping*) [35]. В основе работы данного подхода лежит извлечение ориентиров из информации, получаемой с датчиков системы, — выделяющихся особенностей, относительно которых может вестись оценка положения системы. В качестве таких ориентиров, извлекаемых из датчиков, могут выступать ключевые точки, линии и плоскости. В частности, с повышением доступности датчиков 3D-сканирования плоскости все чаще используются в SLAM-системах [24, 31, 17, 20] для улучшения качества локализации системы в пространстве. Это вызвано рядом причин: во-первых, ориентиров-плоскостей в средах значительно меньше, чем других ориентиров, что значительно сокращает вычислительную сложность задачи, во-вторых, данные объекты в меньшей степени зависимы от выбросов и шума, обусловленных особенностями используемых сенсоров. То, что планарные поверхности повсеместно встречаются в окружающей среде, созданной человеком, является дополнительным аргументом к их активному использованию. Так, внутри помещений примерами могут служить предметы интерьера, пол и стены (рис. 1). В городских средах плоскостями можно считать фасады зданий, дороги.

Наиболее известным для решения задачи сегментации плоскостей является семейство алгоритмов RANSAC [10]. Универсальность метода позволяет использовать его как на разреженных данных, полученных с сенсоров LiDAR [30], так и на данных с камер RGB-D [2, 19], применяющихся в основном внутри помещений. Несмотря на высокую точность, тяжеловесность вычислений [6] затрудняет использование RANSAC в задачах реального времени. Метод разрастания регионов [9, 25, 27] — другой класс алгоритмов, основанный на упорядоченной структуре сним-

(а) Исходный RGB-D снимок

(б) Размеченные плоскости

Рис. 1: Сегментация плоскостей на датасете ICL NUIM [32]

ков с камер глубины. С помощью информации о параметрах камеры возможно установить биекцию между пикселями изображения и точками его 3D-проекции (называемой *облаком точек*).

В силу относительной дешевизны и простоты использования по сравнению с LiDAR, сенсоры RGB-D всё чаще оказываются установленными на автономные системы. Безусловным преимуществом сенсоров RGB-D является их сравнительно невысокая цена, однако такие системы в силу ограниченной дальности работы камер RGB-D предназначены для работы в основном внутри помещений. Из-за популярности подобных сенсоров на SLAM-системах реального времени существует необходимость в высокопроизводительных алгоритмах сегментации плоскостей, способных обрабатывать поступающие в систему кадры с камер RGB-D в режиме реального времени. Исследование современных алгоритмов, представленных в [6], показало, что большинство их реализаций требует существенной доработки для использования в реальных задачах сегментации. К недоработкам можно отнести отсутствие грамотной архитектуры приложения, невозможность параметризации алгоритма и отсутствие удобного способа интеграции в исследовательские Python-проекты.

Отдельно стоит отметить важность доступности алгоритма на автономных системах, где используется связующее программное обеспе-

чение ROS (Robot Operating System) [29] на базе семейства операционных систем Linux. ROS, по сути являющаяся стандартным решением в области робототехники, позволяет как установить передачу сообщений между компонентами системы по разнообразным протоколам, так и использовать множество библиотек (иначе называемых *ROS-пакетами*) для задач обработки изображений, визуализации, управления перемещением робота и т.п.

Таким образом, в настоящее время в открытом доступе не хватает библиотеки, которая бы соответствовала всем требованиям современной разработки программного обеспечения и могла бы использоваться для быстрой сегментации изображений как при прототипировании SLAM-систем на Python, так и для их апробации на реальных автономных системах с помощью ROS пакета. Данная работа посвящена разработке библиотеки сегментации плоскостей на RGB-D изображениях, которая покрывает вышеописанные запросы.

1. Постановка задачи

Цель работы — реализация библиотеки сегментации плоскостей из RGB-D снимков. Для достижения цели были поставлены следующие задачи:

- провести обзор и сравнение алгоритмов сегментации плоскостей и выбрать алгоритм для реализации;
- разработать требования к системе, спроектировать и реализовать библиотеку с выбранным алгоритмом;
- создать Python- и ROS-пакеты для полученной библиотеки;
- оценить точность и время работы реализованного алгоритма сегментации.

2. Обзор

В данном разделе представлен обзор существующих классов алгоритмов, их особенностей и ограничений для выбора алгоритма, который должен быть реализован в конечной библиотеке. В секции также уделено внимание некоторым особенностям программного обеспечения ROS.

2.1. Универсальные методы извлечения плоскостей

Существуют алгоритмы, способные работать независимо от типа установленных на систему сенсоров, что расширяет возможности систем SLAM, но вместе с этим и вносит некоторые ограничения. Ниже приведены основные классы данных алгоритмов.

2.1.1. RANSAC

Большое количество работ в области сегментации облаков точек и изображений ссылаются на семейство методов RANSAC (Random Sample Consensus) [10]. В основе алгоритма лежит перебор математических моделей, которые наиболее хорошо аппроксимируют некоторое подмножество элементов. Благодаря вероятностному перебору большого числа моделей, результаты RANSAC практически независимы от выбросов. Данный подход показывает высокую точность сегментации [30], и может быть применим не только для сегментации плоскостей, но и других геометрических примитивов [3]. Тем не менее, тяжеловесность вычислений [6] затрудняет использование RANSAC в задачах реального времени.

2.1.2. Преобразование Хафа

Преобразование Хафа — другой универсальный подход, использующийся для обнаружения геометрических объектов на изображении. Частным случаем его применения является извлечение плоскостей в 3D пространстве [1, 7]. Для этого устанавливается соответствие между

3D пространством объекта и пространством параметров плоскости, подбираемых с помощью процедуры голосования. Методы из этого семейства алгоритмов применимы как для сегментации плоскостей из LiDAR данных [8], так и из RGB-D изображений [5, 14]. Необходимость тонкой настройки параметров алгоритма и его высокое время работы [9] на зашумленных данных так же ставит под сомнение его использование в современных алгоритмах сегментации плоскостей.

2.2. Разрастание регионов на RGB-D снимках

Рассмотренные выше алгоритмы не предполагают наличия знаний о природе данных. В случае когда задача сегментации ограничивается данными с RGB-D камер, можно использовать информацию об организованной структуре изображений. RGB-D снимок, представляющий собой матрицу со значениями глубины пикселей $p(u, v)$, может быть спроецирован на трёхмерное множество точек $P(x, y, z)$. Это позволяет как итерироваться по удобной двумерной структуре изображения, так и пользоваться преимуществами доступных операций над объектами трёхмерного евклидова пространства.

Одним из подходов, показывающим хорошие результаты по скорости и качеству [25, 9, 6] сегментации RGB-D снимков, является *метод разрастания регионов*. Общая схема работы такого алгоритма приведена на рисунке 2.

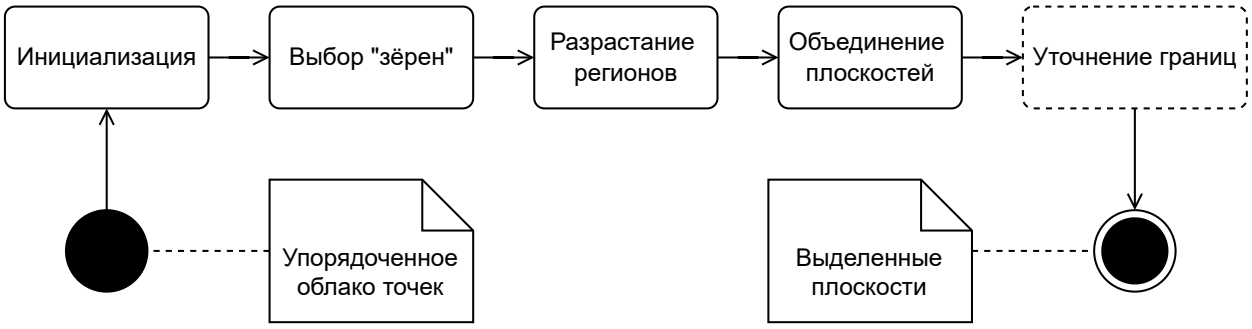

Рис. 2: Общая схема работы алгоритмов разрастания регионов

В основе идеи лежит поиск подходящих под ограничения начальных точек на изображении («зёрен») с последующим их разрастанием

в планарные регионы. В частности, для проверки планарности может использоваться [9, 25] *метод главных компонент (PCA — Principal Component Analysis)* [22].

2.3. Избранные алгоритмы разрастания регионов

В исследовании в работе [6] приведено множество современных алгоритмов сегментации плоскостей на RGB-D снимках. Согласно данному исследованию три наиболее успешных в терминах качества сегментации и скорости работы алгоритма, PEAC [9], CAPE [25] и DDPFF [27] — основаны на методе разрастания регионов и имеют схожую схему работы, показанную на рис. 2, за исключением различной параметризации и технических деталей реализации стадий. В данной секции представлен обзор их алгоритмов и существующих в открытом доступе реализаций.

2.3.1. PEAC

Алгоритм PEAC основывается на идее извлечения линий с помощью агломеративной кластеризации [33], обобщенной на трёхмерное пространство. На стадии *инициализации* изображение разбивается на фрагменты одинакового размера, называемые *клетками*. Каждый фрагмент проверяется на состоятельность: достаточное число ненулевых точек, непрерывность значений глубины и планарность клетки по методу главных компонент. Среди подходящих клеток выбирается одна *клетка-зерно* с наименьшей средней квадратичной ошибкой. На этапе *разрастания регионов*, присутствующего на общей схеме работы (рис. 3), выбранный сегмент расширяется по соседним клеткам до тех пор, пока его значение ошибки не превышает значения соответствующего заданного параметра. Для PEAC разрастание регионов и *объединение плоскостей* считается единым этапом, после которого следует необязательный этап *уточнения границ*. Такие артефакты как пилообразные границы плоскостей, пересегментация или неиспользованные точки решаются в алгоритме путём затирания старых границ плоскостей с их последующем новым разрастанием.

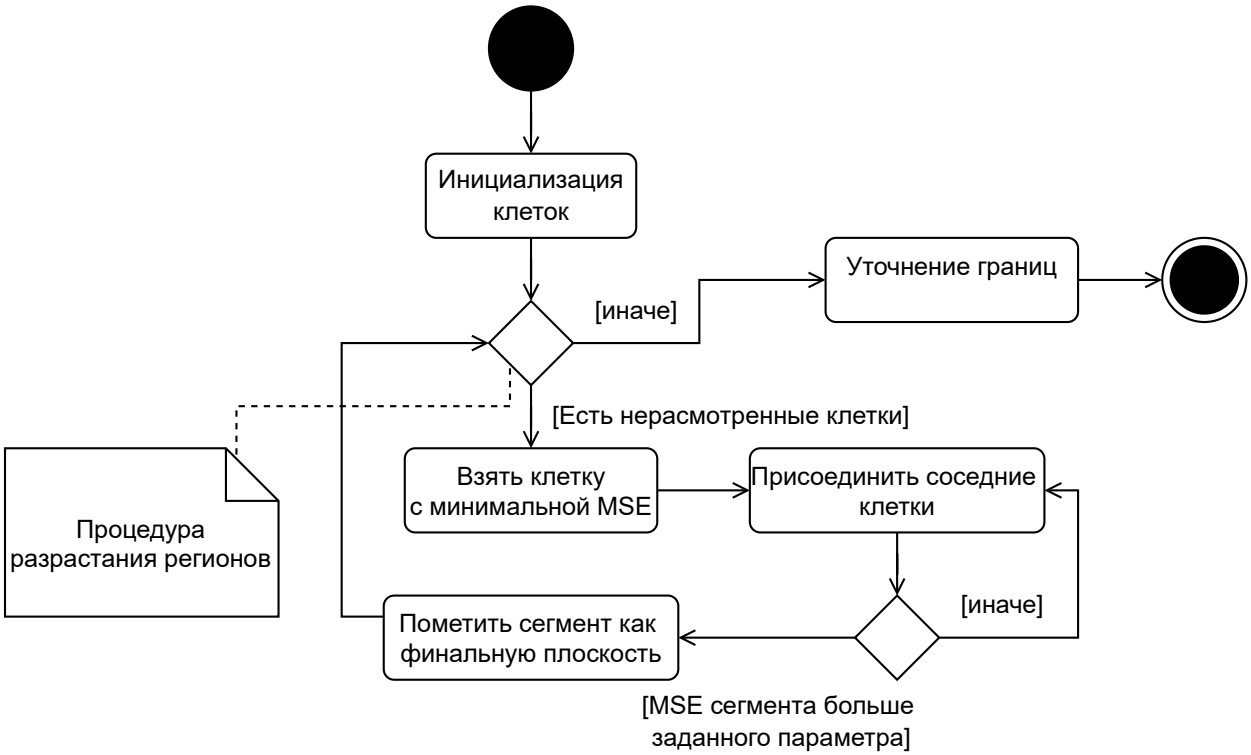

Рис. 3: Диаграмма активностей алгоритма PEAC

Реализация алгоритма PEAC находится в открытом доступе[1]. Стоит отметить, что данная реализация, представленная в виде C++ библиотеки из заголовочных файлов, обладает достаточно хорошей архитектурой, а её алгоритмические решения просматриваются в реализациях прочих более современных методов. К недостаткам можно отнести зависимость PEAC от тяжеловесных библиотек, таких как OpenCV [23] и Point Cloud Library (PCL) [28], а также отсутствие Python и ROS пакетов.

2.3.2. CAPE

Алгоритм CAPE описывается авторами как эффективное расширение PEAC, к тому же позволяющее находить цилиндрические поверхности. Хотя метод извлечения плоскостей идентичен предыдущему алгоритму, есть некоторые различия — клетки-зёрна оптимально выбираются исходя из преобладающего направления нормалей, а не в

[1]PEAC, URL: `https://github.com/ai4ce/peac` (дата обращения: 2023-02-12)

последовательном порядке, как это делается в PEAC. Кроме этого, тяжеловесное разрастание регионов при уточнении границ в CAPE заменено на морфологические операции, за счёт чего CAPE демонстрирует существенный прирост в скорости работы алгоритма [25].

Реализация CAPE также находится в открытом доступе[2]. Из обзора репозитория можно заключить, что данная реализация на языке C++ является скорее исследовательским прототипом, нежели полноценным алгоритмом, готовым для использования в реальных задачах. К недостаткам можно отнести отсутствие какого-либо явного разделения стадий алгоритма на отдельные программные компоненты, использование сырых указателей, невозможность настройки параметров, а также встречающиеся в коде алгоритмические неточности и опечатки.

2.3.3. DDPFF

Depth Dependent Plane Flood Fill следует всё той же общей структуре, описанной на рисунке 2. Существенным отличием от предыдущих алгоритмов является использование множества параметров, принимающих во внимание известные различия в плотности точек в зависимости от их глубины [27]. Кроме этого в DDPFF отсутствует проверка на планарность с помощью PCA, вместо этого клетки изначально соединяются друг с другом исходя из угла между их нормалями и ортогонального расстояния от точек одной плоскости до другой.

Алгоритм DDPFF реализован[3] на C++17 с использованием современных практик разработки. Исходный код сильно завязан на возможностях визуализации, примененных в демонстрационном приложении из репозитория, что влечёт за собой необходимость в самостоятельной модификации исходного кода перед использованием в реальных задачах.

[2]CAPE, URL: `https://github.com/pedropro/CAPE` (дата обращения: 2023-02-12)

[3]DDPFF, URL: `https://github.com/arindamrc/DDPFF` (дата обращения: 2023-02-12)

2.4. Программное обеспечение для автономных систем

В данной секции приведён обзор наиболее популярного в сфере робототехники связующего программного обеспечения Robot Operating System (ROS). В дополнение в секции приведено сравнение с новой версией системы ROS 2.

2.4.1. Программное обеспечение ROS

Как правило, для поддержания внутренних коммуникаций между установленными на систему сенсорами, процессором и прочими компонентами используется специальное программное обеспечение. Robot Operating System или ROS [29] является стандартным решением в сообществе робототехники, предоставляющим большие возможности для взаимодействия со встраиваемыми системами. Реализация обмена сообщениями между компонентами-*узлами* (nodes) сложной системы в зависимости от их характера может быть реализована следующими одним или несколькими способами: *топики* (topics), реализующие модель издатель-подписчик, *сервисы* (services), работающие по принципу запрос-ответ, или *действия* (actions), работающие по тому же принципу, что и сервисы, но предоставляющие возможность получать дополнительную информацию вместе с ответами.

Одним из главных преимуществ ROS является широкий выбор официально поддерживаемых или пользовательских библиотек, именуемых *ROS-пакетами*, предназначенных для таких задач, как обработка изображений, операции с облаками точек, получение информации с установленных на систему сенсоров и управление перемещением автономной системы. За счёт высокого уровня абстракции ROS позволяет в одном приложении использовать пакеты с узлами, написанными как на языке C++, так и на языке Python.

2.4.2. ROS и ROS 2

Несмотря на большую популярность ROS, изначально она разрабатывалась как система для образовательных целей, а потому обладает рядом недостатков [26], затрудняющих дальнейшее её поддержание для решения новых стремительно усложняющихся задач. На замену системе ROS, официальная поддержка которой закончится в 2025 году, активно разрабатывается система ROS 2. Авторы позиционируют её как систему с полностью переработанной архитектурой и подходом к используемым решениям на основе опыта реальных проблем с системой ROS. Некоторые из ключевых отличий, описанных в работе [26], представлены в таблице 1.

Таблица 1: Сравнение ROS 1 и ROS 2

Параметр	**ROS 1**	**ROS 2**
Способ передачи сообщений	Свой протокол, основанный на TCP/UDP	DDS
Поддерживаемые платформы	Linux	Linux, Windows, macOS
Клиентские библиотеки	Реализованы на разных языках	В основе всех лежит C библиотека (rcl)
Узел и процесс	Одному процессу — один узел	Одному процессу — множество узлов
Встраиваемые системы	Минимальная экспериментальная поддержка	Коммерчески поддерживаемая реализация (micro-ROS)

Из-за критических изменений в ROS 2 отсутствует обратная совместимость с предыдущей версией системы, однако настоящий процесс написания приложений и сборки пакетов максимально соответствует предыдущей версии ROS.

3. Сравнение алгоритмов сегментации

Выбранные в ходе обзора алгоритмы PEAC [9], CAPE [25] и DDPFF [27] требуют детального анализа их качества и времени сегментации. В данной секции вместо параметров по умолчанию, используемых в обзоре [6], описан подбор оптимальных параметров для каждого алгоритма как на реальных, так и на синтетических данных.

3.1. Описание данных

Эксперименты с выбранными алгоритмами проводились как на искусственно сгенерированных датасетах ICL [32], так и на реальных изображениях датасета TUM [4].

Таблица 2: Описание датасета

Датасет	Последовательность	Число кадров	Число плоскостей
ICL	living_room	1509	13
	office_room	1510	13
TUM	fr3_cabinet	1119	5
	fr3_long_office_validation	2603	8

Описание датасетов было заимствовано из [6]. Число плоскостей в таблице 2 означает количество уникальных планарных объектов на всей последовательности.

Для оценки качества алгоритмов с помощью пакета evops был взят одноимённый датасет [21] с размеченными последовательностями из таблицы 2.

3.2. Подбор параметров алгоритмов

Для сравнения выбранных методов был реализован инструмент[4], позволяющий не только запускать алгоритмы с различными параметрами и визуализировать результаты сегментации, но и численно сравнивать точность сегментации плоскостей с размеченными ground-truth

[4]Algo-Runner, URL: `https://github.com/prime-slam/algo-runner` (дата обращения: 2023-04-20)

данными. Последнее было реализовано с помощью Python пакета метрик evops [6]. Данный пакет включает в себя как классические для задач семантической сегментации экземпляров (instance segmentation) метрики (precision, recall, noise и т.п.), так и более продвинутые метрики (panoptic), агрегирующие оценку на уровне сущностей (плоскостей) и на уровне отдельных пикселей. Данные метрики подробно рассмотрены в магистерской диссертации студента СПбГУ Яроша Дмитрия Сергеевича «Алгоритмы SLAM с использованием планарных поверхностей». За основу инструмента были взяты алгоритмы, собранные в процессе работы Дмитрия Сергеевича в Docker-контейнеры[5].

Так как алгоритмы предоставляли только параметры по умолчанию, для более корректного сравнения было решено зафиксировать четыре последовательности из RGB-D снимков, описанных в таблице 2, и подобрать под них параметры алгоритмов. В рамках летней школы СПбГУ совместно со студентами 2 курса СПбГУ Киселёвым Михаилом Николаевичем и Ржеусским Богданом Викторовичем были подобраны оптимальные параметры для указанных в таблице последовательностей. Для каждого кадра взятого датасета был посчитан набор значений метрик из пакета evops, в качестве итоговой статистики для значений каждой метрики было взято среднее арифметическое. Результаты на реальных и синтетических данных, соответственно *tum_long_office* и *icl_living_room*, продемонстрированы на рисунке 4. Результаты для всех последовательностей находятся в GitHub[4] репозитории инструмента.

На рисунке 4 представлены алгоритмы PEAC (coarse) и CAPE (coarse) без стадии уточнения границ, а также оригинальный алгоритм DDPFF. Из значений метрик видно, что даже с грубой сегментацией алгоритмы PEAC и CAPE демонстрируют хорошие для задач SLAM результаты. Результаты алгоритма DDPFF сильно зависят от особенностей сцены, что обусловлено его сложной математической моделью, затрудняющей задание подходящих для всей последовательности оптимальных пара-

[5]Plane Detection Dockers, URL: `https://github.com/prime-slam/plane-detection-dockers` (дата обращения: 2022-10-14)

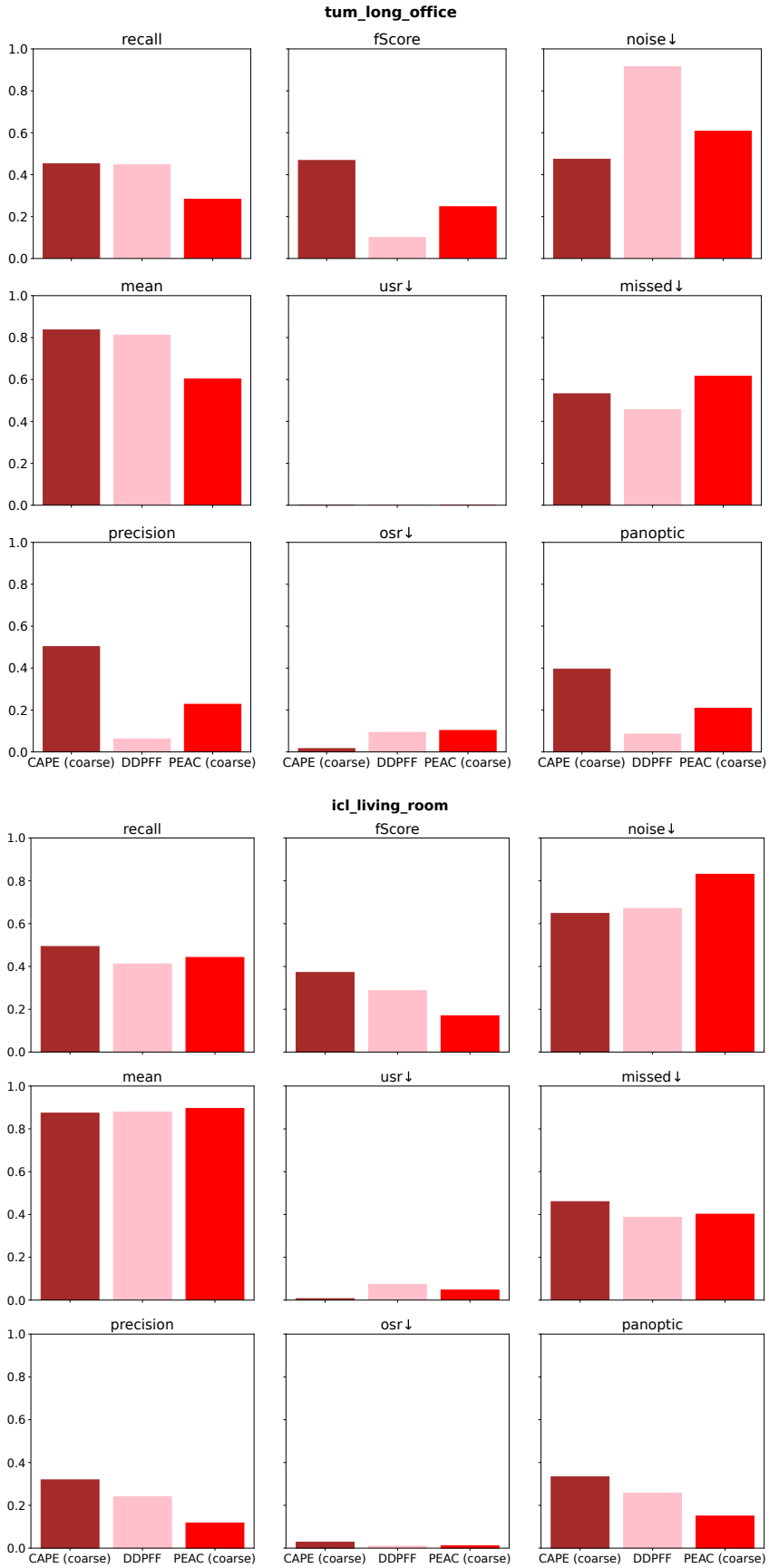

Рис. 4: Сравнение качества алгоритмов сегментации

метров. По этой причине на данном этапе решено отказаться от алгоритма DDPFF в пользу более отзывчивых на настройку параметров алгоритмов PEAC и CAPE.

Стоит отметить, что в алгоритмах SLAM с плоскостями, подаваемые на вход плоскости не должны содержать лишних пикселей, не принадлежащих плоскости, поскольку это приведет к ошибке в вычислении уравнения плоскости, и как следствие ошибке оценки положения.

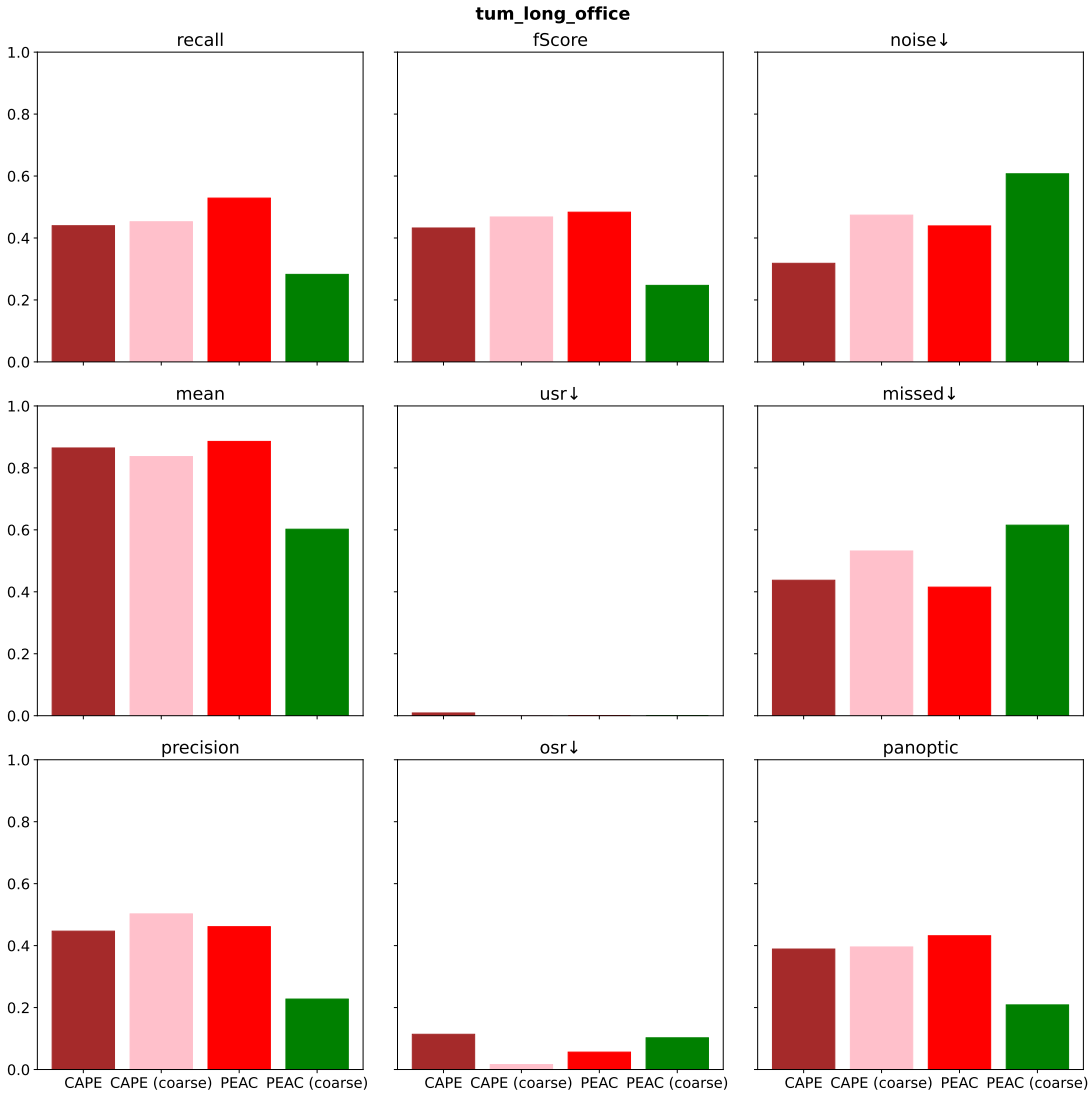

Рис. 5: Алгоритмы со стадией уточнения границ

Отдельно стоит рассмотреть алгоритмы PEAC и CAPE в условиях грубой сегментации и с отдельной стадией уточнения границ плоскостей (рис. 5). Отметим, что так как в алгоритмах PEAC и CAPE стадия уточнения границ реализована по-разному, то её использование приво-

дит к разным результатам. Так, «улучшенный» алгоритм CAPE обладает меньшим шумом, но большим отношением пересегментации (osr), что обусловлено особенностями используемой в алгоритме морфологической обработки границ плоскостей. В свою очередь, PEAC со стадией уточнения показывает стабильно хорошие результаты, значительно превосходящие грубую сегментацию. Вызвано это повторным применением надёжного, но тяжеловесного разрастания регионов на границах плоскостей. Сравнительно хорошие результаты по большинству метрик на графике показывает алгоритм CAPE с грубой сегментацией.

4. Реализация библиотеки

Библиотека сегментации плоскостей должна быть реализована согласно современным принципам разработки программного обеспечения. С учётом сложности предметной области необходим аккуратный подход к разработке требований при проектировании библиотеки и реализации алгоритма сегментации. В данной секции описано проектирование библиотеки deplex, а также особенности реализованного в ней алгоритма сегментации.

4.1. Проектирование библиотеки

Для реализации библиотеки сегментации плоскостей необходимо принимать во внимание множество факторов: от поддержания возможности удобной модификации разрабатываемого алгоритма сегментации, до особенностей работы встраиваемых систем реального времени. В данной секции сформулированы требования к разрабатываемой системе с учётом проведённого ранее в работе обзора других алгоритмов и особенностей целевых сценариев использования.

4.1.1. Разработка требований

Проектируемая библиотека должна быть в первую очередь применима в системах реального времени SLAM, где изображения с камеры глубины поступают в систему с частотой примерно 30 кадров в секунду (FPS). Кроме этого, алгоритм сегментации должен обладать достаточным качеством с низким отношением пересегментации, так как в основном для работы алгоритмов SLAM необходимы большие плоскости. Принимая во внимание сложность алгоритма сегментации, необходимо проектировать архитектуру приложения так, чтобы было возможным вносить изменения в отдельные стадии алгоритма, не затрагивая при этом другие компоненты системы. Для удобного прототипирования разрабатываемая библиотека должна быть доступна широкому кругу пользователей на языке программирования Python. Важным являет-

ся возможность использования библиотеки во встраиваемых системах, работающих преимущественно на операционной системе ROS [29].

Таким образом, к нефункциональным требованиям относятся:

- обработка изображений с частотой не менее 30 FPS;
- низкое значение метрики пересегментации (osr);
- возможность работы с языком Python;
- возможность работы в ROS приложениях.

Принимая во внимание разнообразные модели камер и среды работы автономной системы, алгоритм должен корректно отрабатывать пользовательские параметры. Так, к функциональным требованиям можно отнести возможность запуска алгоритма с пользовательскими параметрами.

4.1.2. Архитектура библиотеки

Для реализации библиотеки был выбран язык программирования C++ по следующим причинам: с использованием низкоуровневых возможностей языка возможно добиться необходимой скорости алгоритма, подходящей для работы в системах реального времени; кроме этого язык C++ является нативным для разработки пакетов для операционной системы ROS [29], на которой работает большинство встраиваемых систем SLAM. Для совместимости ABI библиотеки с другими приложениями, разработка велась на стандарте языка C++14, являющемся стандартом по умолчанию для ROS.

Так как алгоритм предполагает повсеместную работу с математическими структурами и операциями, для этих целей была использована C++ библиотека Eigen [13], предоставляющая широкие возможности высокопроизводительных операций над векторами и матрицами.

Библиотека проектировалась с учётом её основных сценариев использования: C++ приложения, фреймворки SLAM и Python приложения. Диаграмма компонентов библиотеки представлена на рисунке 6.

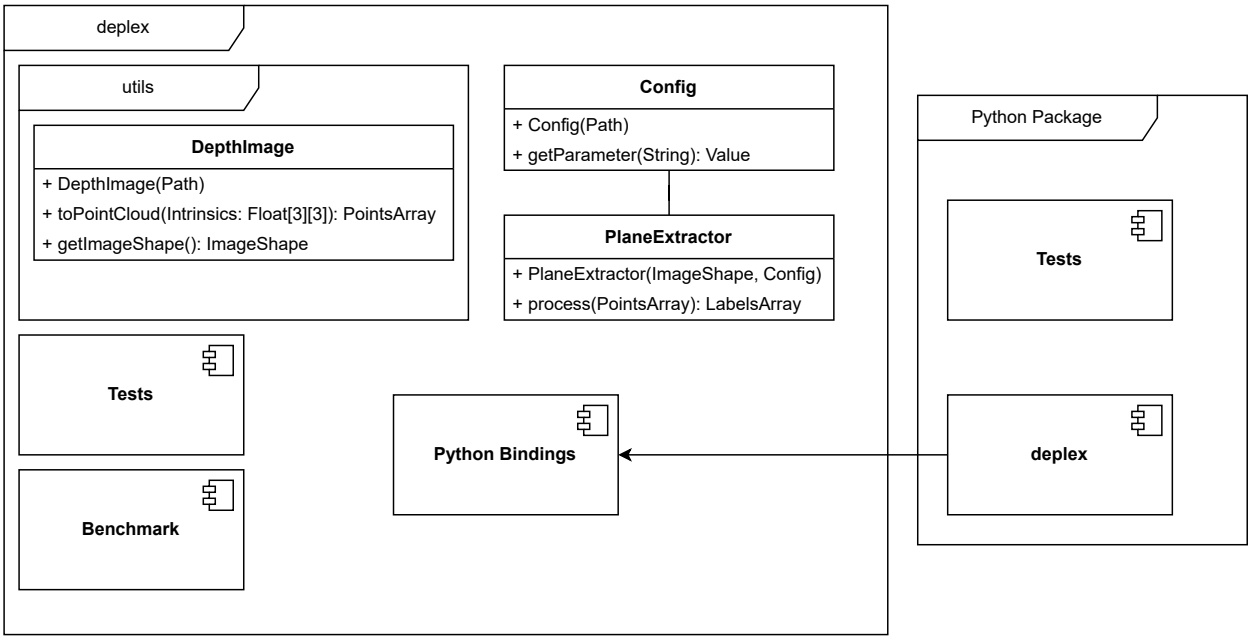

Рис. 6: Диаграмма компонентов библиотеки

Класс `PlaneExtractor` и ассоциированный с ним `Config` отвечают за алгоритм сегментации и его параметризацию, они же являются публичным API библиотеки. Класс `DepthImage` необходим для чтения изображения и преобразования его в облако точек, и так же является частью публичного API. Компоненты `Tests` и `Benchmark` пакета `deplex` являются соответственно C++ тестами и бенчмарком алгоритма, реализованными с помощью GoogleTest [15] и Google Benchmark [16]. Для поддержки использования библиотеки в языке Python были также написаны обвязки для C++ кода, представленные компонентой `Python Bindings`.

Python часть библиотеки, отображенная на диаграмме 6 пакетом `Python package`, так же включает отдельные тесты, реализованные на фреймворке PyTest [34].

4.2. Реализация алгоритма сегментации

За основу алгоритма сегментации были взяты идеи из CAPE, как оптимальное решение в терминах качества результатов (рис. 4) и времени работы [25]. К поставленным целям при реализации библиотеки относилось поддержание простого публичного API, не завязанного на приватных header-файлах с деталями реализации, а также гибкость

структуры для будущих модификаций с целью улучшения скорости работы.

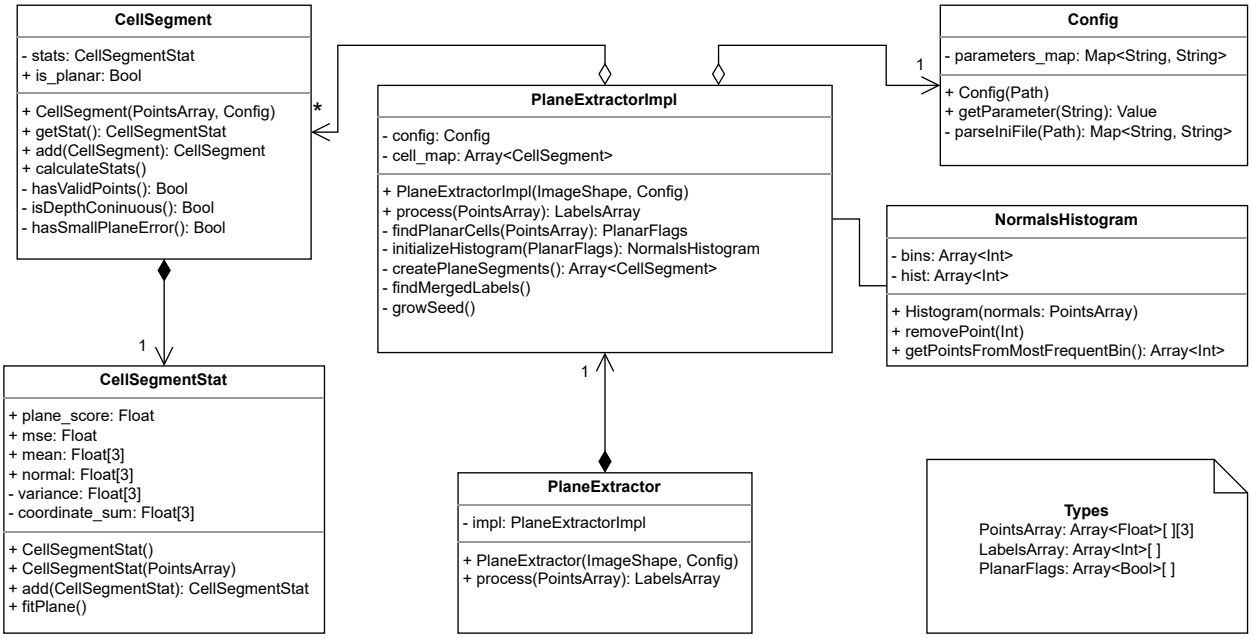

Рис. 7: Диаграмма классов библиотеки

4.2.1. PlaneExtractor

Представленные на диаграмме 7 классы `PlaneExtractor` и `Config` являются публичным интерфейсом библиотеки. Для сокрытия деталей реализации и решения проблемы связности заголовочных файлов в публичном интерфейсе была применена идиома PIMPL (Pointer to IMPLementation) [12], реализованная классом `PlaneExtractorImpl`. Приватные методы класса `PlaneExtractorImpl` соответствуют стадиям, представленным на рисунке 2, и реализуют идеи алгоритма CAPE.

4.2.2. CellSegment

Классом `CellSegment` представлены сегменты из клеток, каждая из которых является фиксированно малого размера матрицей подмножества пикселей исходного изображения.

На стадии инициализации (рис. 2) для каждого начального сегмента, представленного единственной клеткой, проверяются необходимые

условия для планарности. Проверки реализованы в следующем порядке: проверка на достаточность числа ненулевых точек, проверка непрерывности значений глубины между соседними пикселями и измерение ошибки плоскости с помощью PCA.

Логика работы со статистиками сегментов вынесена в композитный класс `CellSegmentStat`. В методе `fitPlane`() реализован PCA с эффективным итеративным обновлением при поступлении новых точек в сегмент, поиск собственных чисел выполнен с помощью быстрого гибридного алгоритма для матриц 3×3 [18]. Критерий планарности определяется соотношением собственных чисел матрицы ковариации точек.

4.2.3. NormalsHistogram

Класс `NormalsHistogram` строит гистограмму нормалей в полярной системе координат для планарных клеточных сегментов. Доминирующее направление нормалей является критерием выбора клетки для разрастания региона на стадии выбора «зёрен» (рис. 2).

4.2.4. Config

Главное предназначение класса-обёртки `Config` — задание параметров работы алгоритма. Список параметров с их описанием представлен в таблице 3.

В указанной таблице отсутствуют параметры, относящиеся к стадии уточнения границ из рис. 2, так как на текущем этапе работы реализована только стадия грубой сегментации, что избавляет проект от прочих зависимостей, кроме библиотеки Eigen.

4.2.5. Ключевые различия в реализации CAPE

Основным отличием по сравнению с оригинальной реализацией является большая модульность проекта, что позволяет легко модифицировать отдельные стадии алгоритма, возможность удобной параметризации. Кроме этого в библиотеке deplex исправлены многие алгоритмические ошибки, присущие оригинальной версии.

Таблица 3: Параметры алгоритма PlaneExtractor

Имя параметра	Описание
	Инициализация
patchSize	Размер изначальных клеток
minPtsPerCell	Минимальное число ненулевых точек, чтобы считать клетку валидной
depthDiscontinuityThreshold	Максимальная разность значений глубины, при которой соседние точки считаются непрерывными
maxNumberDepthDiscontinuity	Максимальное число разрывов
depthSigmaCoeff, depthSigmaMargin	Коэффициенты оценки ошибки плоскости
	Выбор «зёрен»
histogramBinsPerCoord	Степень зернистости оценки направлений нормали
minRegionGrowingCandidateSize	Минимальное число нормалей клеток, чтобы считать направление преобладающим
	Разрастание регионов
minRegionGrowingCellsActivated	Минимальное число клеток-соседей, для принятия сегмента в рассмотрение
minRegionPlanarityScore	Минимальное значение оценки планарности, при котором сегмент считается финальной плоскостью
	Объединение плоскостей
minCosAngleForMerge	Максимальное значение угла между плоскостями для их объединения
maxMergeDist	Максимальное расстояние между плоскостями для их объединения

Другим важным отличием является отсутствие последней стадии уточнения границ сегментации. Обусловлено это тем, что для алгоритмов SLAM нам не так важны границы плоскостей. Из-за высокой модульности реализованной библиотеки при необходимости в дальнейшем алгоритм может быть расширен данной стадией.

4.3. Создание Python пакета

Реализованная библиотека deplex должна быть доступна широкому кругу пользователей для удобной установки и использования в том числе при разнообразном прототипировании на языке Python. Для достижения поставленной задачи было решено собрать из реализованной библиотеки pip-пакет и сделать его доступным на PyPI, индексе пакетов Python. Инструменты для сборки кросс-платформенного пакета были

выбраны на основе обзора в собственной производственной практике «Безреференсная метрика для оценки качества траектории из облаков точек».

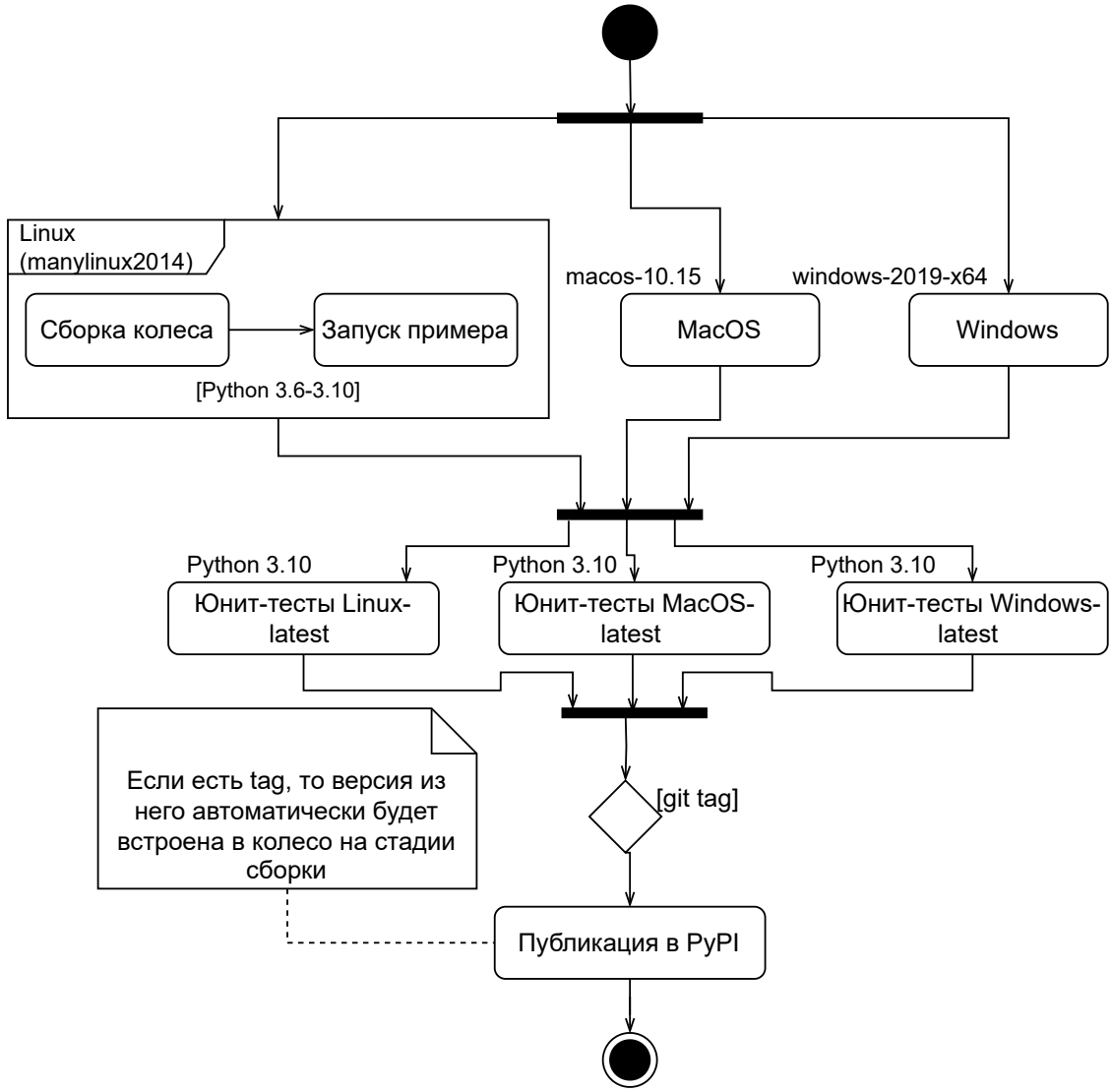

Рис. 8: Сборка и публикация пакета в PyPI

Автоматическая сборка и публикация пакета реализованы с помощью GitHub Actions [11] и представлены на диаграмме 8. Собранный Python-пакет находится в индексе PyPI[6].

[6]deplex PyPI, URL: `https://test.pypi.org/project/deplex/` (дата обращения: 2023-03-12)

4.4. Создание ROS пакета

Для апробации библиотеки был написан ROS 2 пакет на языке программирования C++, а также демо-приложение, имитирующее работу реальной автономной системы.

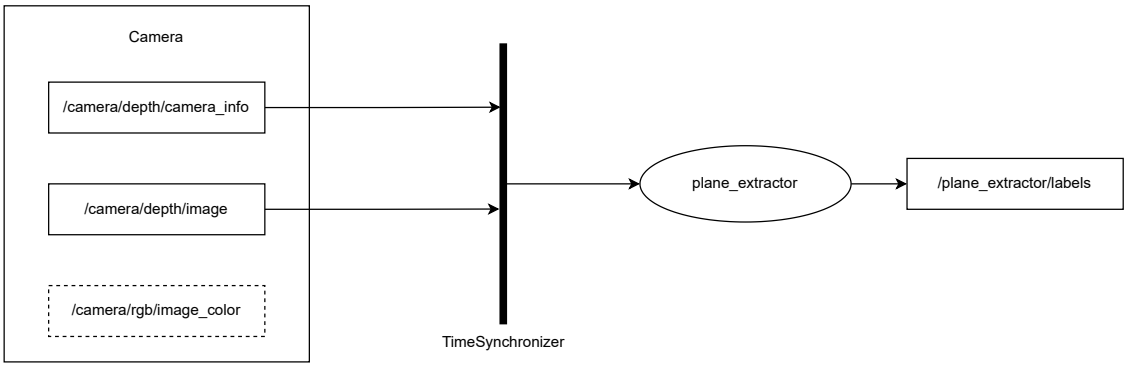

Рис. 9: ROS-граф демо-приложения

На рисунке 9 изображен ROS-граф работы демонстрационного приложения, использующего реализованную библиотеку. Алгоритм deplex представлен *узлом* `plane_extractor`. Для работы приложения, узел `plane_extractor` подписывается на синхронизированные по времени на соответствующие кадрам сообщения из *топиков* `camera_info` и `image` — соответственно параметры калибровки камеры глубины и маска глубины. Для приближения к настоящим условиям была взята последовательность TUM из таблицы 2 в формате *rosbag*, хранящая в себе сообщения с топиков реальной камеры. После обработки изображения, данные в виде массива меток публикуются в топик `labels`, на который могут подписаться компоненты сложного приложения.

Рис. 10: Визуальное сравнение качества сегментации

5. Эксперимент

Реализованный алгоритм сегментации нуждается в численном сравнении точности с оригинальной реализацией алгоритма CAPE. Анализ текущей точности алгоритма необходим для формулирования дальнейших целей его модификации.

5.1. Анализ точности сегментации

Задачей эксперимента является анализ качества работы реализованного в данной работе алгоритма сегментации и алгоритма CAPE. Описание данных и инструмента, с помощью которого производился запуск алгоритмов, были приведены ранее в разделе «Подбор параметров алгоритмов».

Для эксперимента были взяты следующие алгоритмы: CAPE и PEAC без стадии уточнения границ, и реализованный алгоритм deplex. Выбранные алгоритмы были последовательно запущены на синтетической последовательности ICL *living_room* и на реальных данных TUM *fr3_long_office_validation* из таблицы 2 с соответственно оптимизированными параметрами, точность на каждом кадре была замерена с помощью пакета метрик evops. В качестве статистик для каждой последовательности посчитаны средние значения для каждой метрики на всех кадрах.

По результатам сравнения, приведённым на рисунке 11, можно отметить, что качество реализованного алгоритма (столбец deplex) в среднем сопоставимо с качеством CAPE. Для лучшего понимания результатов пример сегментации алгоритмов на первом кадре последовательности ICL *living_room* приведён на рисунке 10.

Ещё раз отметим, что для SLAM алгоритмов достаточно грубой сегментации. В то же время из рисунка 11 видно, что качество полученного алгоритма не уступает алгоритмам CAPE и PEAC с грубой сегментацией.

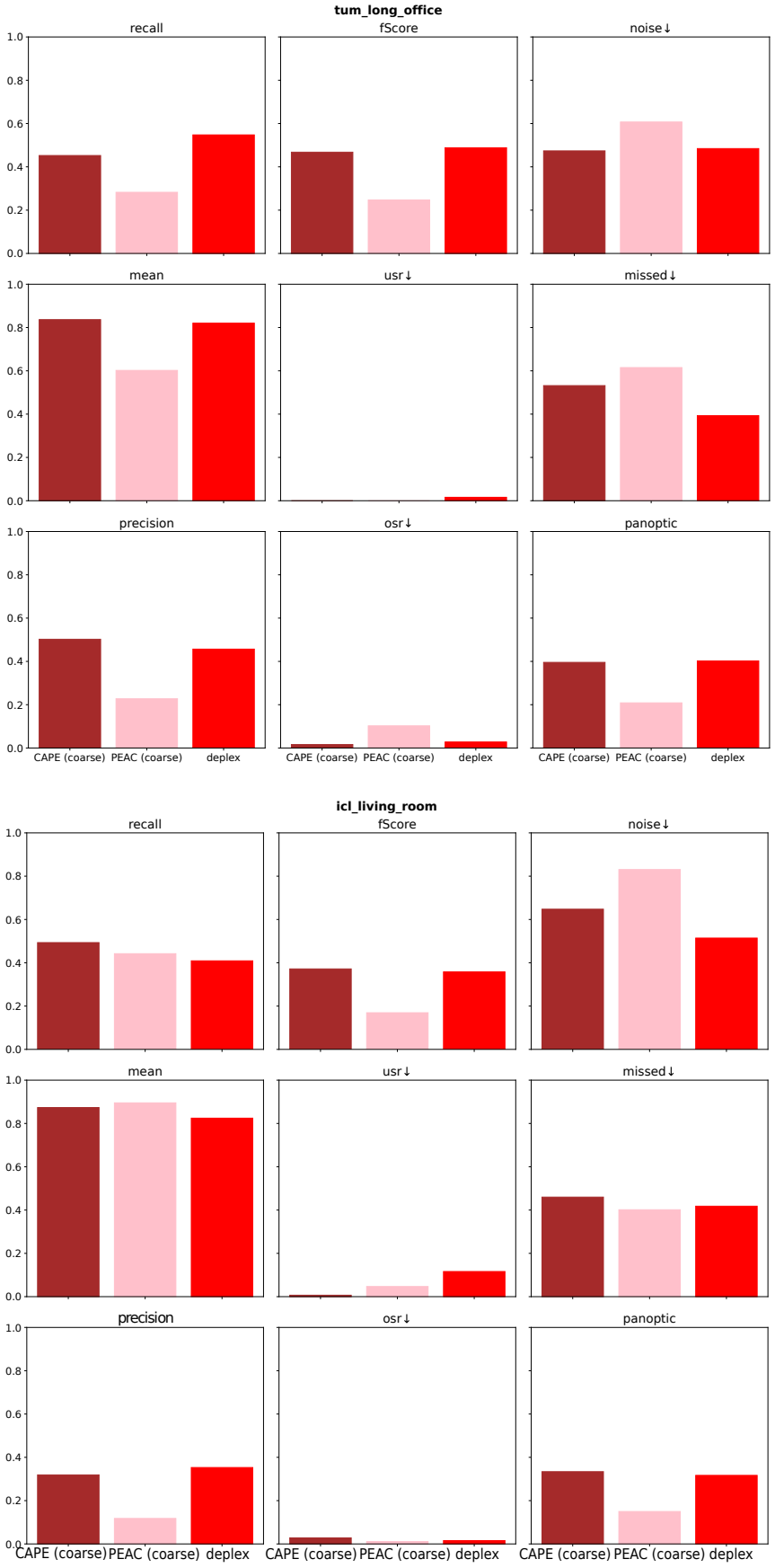

Рис. 11: Сравнение качества deplex

5.2. Анализ времени работы

Большинство SLAM систем реального времени работают с камерами, способными выдавать изображения с разрешением 640×480 пикселей с частотой 30 кадров в секунду (FPS), что уже накладывает минимально необходимое нефункциональное требование к разрабатываемому алгоритму. Кроме этого, важно показать, что скорость алгоритма сопоставима со скоростью оригинальной реализации взятого за основу CAPE.

Для анализа времени работы были зафиксированы следующие характеристики среды выполнения: операционная система Linux Ubuntu 22.04.2 64-bit; аппаратная конфигурация: Intel(R) Core(TM) i5-1135G7 CPU @ 2.40GHz, 8GB RAM. Оба алгоритма скомпилированы GCC 11.3 с уровнем оптимизации O3. В качестве тестовых данных были взяты 1000 последовательных кадров из датасета TUM *fr3_long_office_validation*.

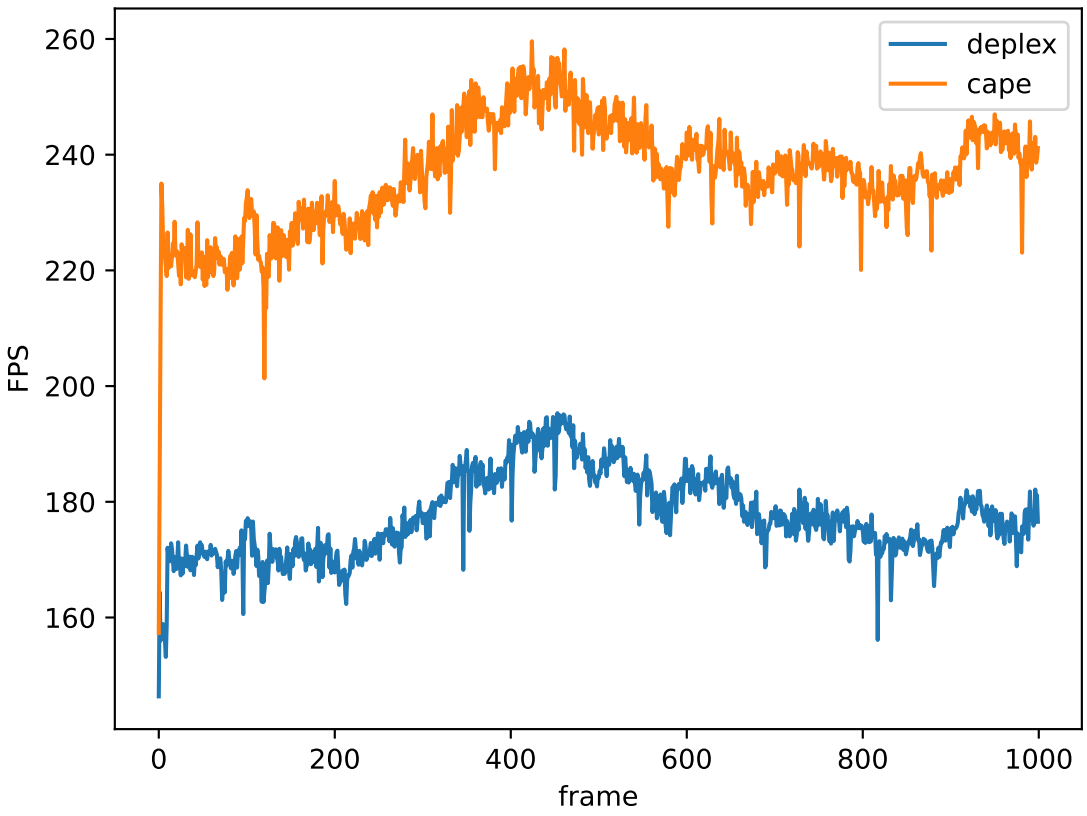

Рис. 12: Сравнение скорости работы deplex и CAPE

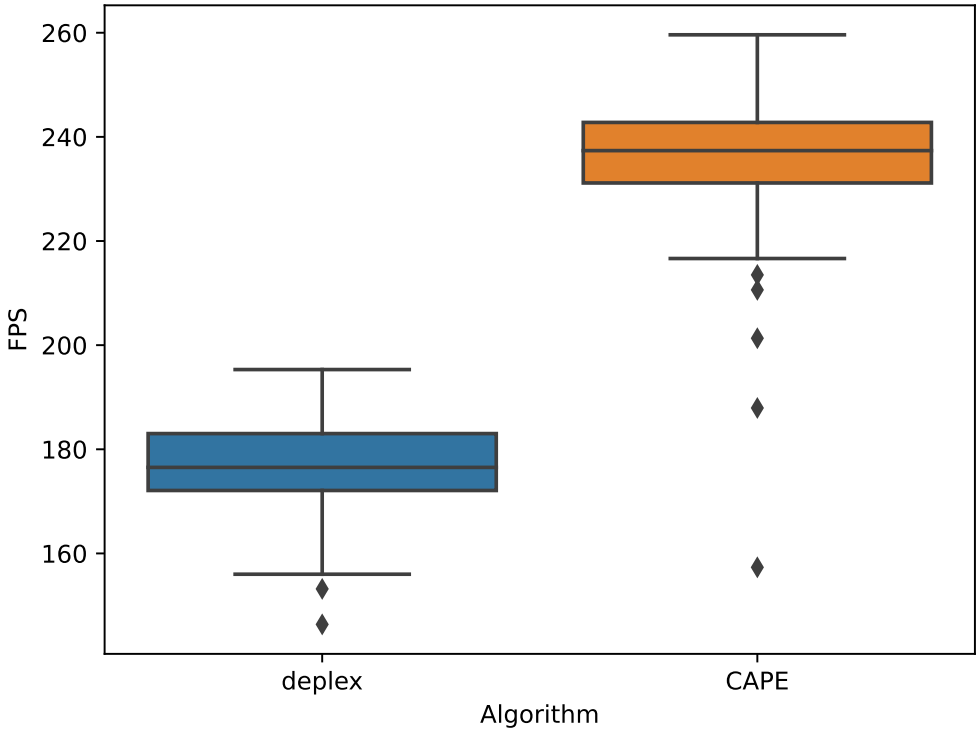

Рис. 13: Распределение частот кадров deplex и CAPE

Отметим, что требование к способности алгоритма обрабатывать 30 кадров в секунду выполняется у обоих алгоритмов. Тем не менее, скорость алгоритма CAPE превышает реализованный в deplex алгоритм на 60 кадров. Различие обусловлено в первую очередь повышенными накладными расходами из-за более аккуратной архитектуры библиотеки и выборе надежных структур данных.

Заключение

В результате работы были выполнены следующие задачи:

- выполнен обзор и сравнение алгоритмов сегментации плоскостей, для реализации выбран алгоритм CAPE;
- спроектирована и реализована библиотека[7] с выбранным алгоритмом на языке C++ с учётом разработанных требований;
- для упрощенного пользования библиотекой были созданы ROS-пакет и Python-пакет[8] для рекомендованных стандартом версий (и дистрибутивов) операционных систем Windows, macOS и Linux;
- проведены эксперименты для оценки точности реализованного алгоритма с помощью пакета метрик evops, а также скорости работы алгоритма. Установлено, что реализованный алгоритм отвечает всем требованиям к системе.

[7]deplex, URL: `https://github.com/prime-slam/deplex` (дата обращения 2023-05-02)

[8]deplex PyPI, URL: `https://test.pypi.org/project/deplex/` (дата обращения 2023-05-02)

Список литературы

[1] Borrmann Dorit, Elseberg Jan, Lingemann Kai, and Nüchter Andreas. The 3d hough transform for plane detection in point clouds: A review and a new accumulator design // 3D Research. — 2011. — Vol. 2, no. 2. — P. 1–13.

[2] Alehdaghi Mahdi, Esfahani Mahdi Abolfazli, and Harati Ahad. Parallel RANSAC: Speeding up plane extraction in RGBD image sequences using GPU // 2015 5th International Conference on Computer and Knowledge Engineering (ICCKE) / IEEE. — 2015. — P. 295–300.

[3] Bazin Jean Charles and Pollefeys Marc. 3-line RANSAC for orthogonal vanishing point detection // 2012 IEEE/RSJ International Conference on Intelligent Robots and Systems. — 2012. — P. 4282–4287.

[4] Sturm J., Engelhard N., Endres F., Burgard W., and Cremers D. A Benchmark for the Evaluation of RGB-D SLAM Systems // Proc. of the International Conference on Intelligent Robot Systems (IROS). — 2012.

[5] Hulik Rostislav, Spanel Michal, Smrz Pavel, and Materna Zdenek. Continuous plane detection in point-cloud data based on 3D Hough Transform // Journal of visual communication and image representation. — 2014. — Vol. 25, no. 1. — P. 86–97.

[6] Kornilova Anastasiia, Iarosh Dmitrii, Kukushkin Denis, Goncharov Nikolai, Mokeev Pavel, Saliou Arthur, and Ferrer Gonzalo. EVOPS Benchmark: Evaluation of Plane Segmentation from RGBD and LiDAR Data. — 2022. — Access mode: `https://arxiv.org/abs/2204.05799`.

[7] Gaspers Bastian, Stückler Jörg, Welle Jochen, Schulz Dirk, and Behnke Sven. Efficient Multi-resolution Plane Segmentation of 3D Point Clouds. — 2011. — 12. — P. 145–156.

[8] Tian Yifei, Song Wei, Chen Long, Sung Yunsick, Kwak Jeonghoon, and Sun Su. Fast planar detection system using a GPU-based 3D Hough transform for LiDAR point clouds // Applied Sciences. — 2020. — Vol. 10, no. 5. — P. 1744.

[9] Feng Chen, Taguchi Yuichi, and Kamat Vineet R. Fast plane extraction in organized point clouds using agglomerative hierarchical clustering // 2014 IEEE International Conference on Robotics and Automation (ICRA). — 2014. — P. 6218–6225.

[10] Fischler Martin A. and Bolles Robert C. Random Sample Consensus: A Paradigm for Model Fitting with Applications to Image Analysis and Automated Cartography // Commun. ACM. — 1981. — jun. — Vol. 24, no. 6. — P. 381–395. — Access mode: https://doi.org/10.1145/358669.358692.

[11] GitHub Actions. — https://docs.github.com/en/actions. — Accessed: 2022-12-17.

[12] Griffiths Alan and Radford Mark. Separating Interface and Implementation in C++. — http://www.octopull.co.uk/c++/implementation_hiding.html. — Accessed: 2022-11-15.

[13] Guennebaud Gaël, Jacob Benoît, et al. Eigen v3. — http://eigen.tuxfamily.org. — 2010.

[14] Vera Eduardo, Lucio Djalma, Fernandes Leandro AF, and Velho Luiz. Hough Transform for real-time plane detection in depth images // Pattern Recognition Letters. — 2018. — Vol. 103. — P. 8–15.

[15] Inc. Google. GoogleTest - Google Testing and Mocking Framework. — https://github.com/google/googletest. — Accessed: 2022-11-15.

[16] Inc. Google. A microbenchmark support library. — https://github.com/google/benchmark. — Accessed: 2022-11-15.

[17] Kaess Michael. Simultaneous localization and mapping with infinite planes // Proceedings - IEEE International Conference on Robotics and Automation. — 2015. — 06. — Vol. 2015. — P. 4605–4611.

[18] Kopp Joachim. Efficient numerical diaganalization of hermitian 3 × 3 matrices // International Journal of Modern Physics C. — 2008. — mar. — Vol. 19, no. 03. — P. 523–548. — Access mode: https://doi.org/10.1142%2Fs0129183108012303.

[19] Kurban Rifat, Skuka Florenc, and Bozpolat Hakki. Plane segmentation of kinect point clouds using RANSAC // The 7th international conference on information technology. — 2015. — P. 545–551.

[20] Geneva Patrick, Eckenhoff Kevin, Yang Yulin, and Huang Guoquan. LIPS: LiDAR-Inertial 3D Plane SLAM. — Madrid, Spain : IEEE Press. — 2018. — P. 123–130. — Access mode: https://doi.org/10.1109/IROS.2018.8594463.

[21] Mokeev Pavel. EVOPS: Evaluation Of Plane Segmentation. Dataset. — https://evops.netlify.app/dataset/. — Accessed: 2022-08-30.

[22] Nurunnabi Abdul, Belton David, and West Geoff. Robust segmentation for multiple planar surface extraction in laser scanning 3D point cloud data // Proceedings of the 21st International Conference on Pattern Recognition (ICPR2012). — 2012. — P. 1367–1370.

[23] OpenCV. Open Source Computer Vision Library. — https://github.com/opencv/opencv. — 2015.

[24] Zhang Xiaoyu, Wang Wei, Xianyu Qi, Ziwei Liao, and Wei Ran. Point-Plane SLAM Using Supposed Planes for Indoor Environments // Sensors. — 2019. — 09. — Vol. 19. — P. 3795.

[25] Proença Pedro F. and Gao Yang. Fast Cylinder and Plane Extraction from Depth Cameras for Visual Odometry // 2018 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS). — 2018. — P. 6813–6820.

[26] Macenski Steven, Foote Tully, Gerkey Brian, Lalancette Chris, and Woodall William. Robot Operating System 2: Design, architecture, and uses in the wild // Science Robotics. — 2022. — Vol. 7, no. 66. — P. eabm6074. — Access mode: https://www.science.org/doi/abs/10.1126/scirobotics.abm6074.

[27] Roychoudhury Arindam, Missura Marceli, and Bennewitz Maren. Plane Segmentation Using Depth-Dependent Flood Fill // 2021 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS). — 2021. — P. 2210–2216.

[28] Rusu Radu and Cousins Steve. 3D is here: Point cloud library (PCL). — 2011. — 05.

[29] Stanford Artificial Intelligence Laboratory et al. Robotic Operating System. — 2018. — Access mode: https://www.ros.org.

[30] Tarsha-Kurdi Fayez, Landes Tania, and Grussenmeyer Pierre. Hough-transform and extended ransac algorithms for automatic detection of 3d building roof planes from lidar data // ISPRS Workshop on Laser Scanning 2007 and SilviLaser 2007. — 2007. — Vol. 36. — P. 407–412.

[31] Trevor Alexander J. B., Rogers John G., and Christensen Henrik I. Planar surface SLAM with 3D and 2D sensors // 2012 IEEE International Conference on Robotics and Automation. — 2012. — P. 3041–3048.

[32] Handa Ankur, Whelan Thomas, McDonald John, and Davison Andrew J. A benchmark for RGB-D visual odometry, 3D reconstruction and SLAM // 2014 IEEE International Conference on Robotics and Automation (ICRA). — 2014. — P. 1524–1531.

[33] Nguyen Viet, Martinelli Agostino, Tomatis Nicola, and Siegwart Roland. A comparison of line extraction algorithms using 2D laser rangefinder for indoor mobile robotics // 2005 IEEE/RSJ International Conference on Intelligent Robots and Systems / IEEE. — 2005. — P. 1929–1934.

[34] Krekel Holger, Oliveira Bruno, Pfannschmidt Ronny, Bruynooghe Floris, Laugher Brianna, and Bruhin Florian. pytest 7.2. — 2004. — Access mode: `https://github.com/pytest-dev/pytest`.

[35] Grisetti Giorgio, Kümmerle Rainer, Stachniss Cyrill, and Burgard Wolfram. A tutorial on graph-based SLAM // IEEE Intelligent Transportation Systems Magazine. — 2010. — Vol. 2, no. 4. — P. 31–43.