Санкт-Петербургский государственный университет

Кафедра системного программирования
Программная инженерия

Салью Артур Кристофович

Безреференсная метрика для оценки качества траектории из облаков точек

Отчёт по производственной практике

Научный руководитель:
ст. преп. кафедры СП Я. А. Кириленко

Консультант:
инженер-исследователь, Сколтех, А. В. Корнилова

Санкт-Петербург
2022

Оглавление

Введение

В настоящее время стремительно развивается область алгоритмов компьютерного зрения, в частности в мобильных автономных системах. Особый интерес представляют собой алгоритмы одометрии и алгоритмы одновременной локализации и построении карты (SLAM) [9, 13]. Перед исследователями и разработчиками данных алгоритмов стоит задача оценки их качества. Для осуществления этой цели существуют метрики, которым необходимы референсные данные, полученные с различных сенсоров мобильных автономных систем.

Классические метрики, такие как ATE (Absolute Trajectory Error) или RPE (Relative Pose Error) [7], основаны на использовании референсных данных, полученных из высокоточных сенсоров. К референсным данным относится, например, траектория, показываемая системами GNSS/INS. Однако на практике оказывается, что данный подход имеет ряд ограничений, которые отрицательно сказываются на оценке качества алгоритма. Так, упомянутые выше системы GNSS/INS обладают погрешностью [4], пренебрежительной лишь на больших расстояниях, тогда как в помещениях чаще используется метод MoCap.

Такие ограничения послужили толчком для развития новых инструментов для оценки качества траекторий, *безреференсных метрик*, которые не нуждаются в референсных данных. Подобные алгоритмы основаны на численном измерении шумов и других дефектов *карты* — набора множеств точек пространства, расположенных согласованно с траекторией. Множество точек пространства по-другому зовётся *облаком точек* и является артефактом работы таких сенсоров как 3D-сканеры местности или оптические «радары» — лидары, являющиеся крайне популярными на автономных системах.

К базовым безреференсным метрикам относятся MPV (Mean Plane Variance) [3] и MME (Mean Map Entropy) [2]. MPV основывается на предположении, что окружающее пространство в большинстве своём представлено плоскостями. У MME же в принципе работы лежит вычисление энтропии карты из облаков точек. Несмотря на свою попу-

лярность, данные метрики обладают слабой корреляцией [6] с ошибкой траектории.

Исследователями из Mobile Robotics Lab, Skoltech была разработана новая безреференсная метрика — MOM (Mutually Orthogonal Metric) [6]. MOM извлекает подмножество точек, лежащих на взаимно ортогональных плоскостях, и затем применяет на нём MPV. В своей статье авторы продемонстрировали лучшую корреляцию данного подхода с ошибкой траектории.

В настоящее время не существует кросс-платформенного инструмента, который агрегировал бы в себе описанные метрики. Текущая версия библиотеки map-metrics[1], написанной на языке Python, объединяет в себе безреференсные методы оценки качества траектории. Тем не менее map-metrics имеет низкую производительность, что делает нецелесообразным использование библиотеки для вычислений в реальном времени. Более того выбранный язык разработки не позволяет применять библиотеку во встраиваемых системах.

В данном отчёте будет рассмотрена реализация безреференсных метрик на языке программирования C++ и создание pip-пакета для удобного их использования на различных платформах.

[1] `https://pypi.org/project/map-metrics/0.0.4`

1. Постановка задачи

Цель работы — реализовать кроссплатформенную библиотеку безреференсных метрик для оценки качества траекторий на языке C++ с Python интерфейсом.

Для достижения поставленной цели на данном этапе выделены следующие задачи:

- провести обзор метрик для оценки качества траектории;
- реализовать библиотеку метрик на языке программирования C++;
- добавить возможность задавать параметры алгоритмов в зависимости от сценария использования датчиков;
- поддержать Python обвязку и CI/CD pip-пакета в индекс PyPI;
- оценить производительность и качество реализованных алгоритмов.

2. Обзор предметной области

В данном разделе проведён обзор наиболее популярных методов оценки алгоритмов одометрии, в том числе безреференсных метрик. Понимание их теоретических основ способствует эффективной программной реализации в рамках библиотеки map-metrics.

2.1. Референсные методы оценки траектории

Для оценки качества траектории наиболее прямолинейным подходом является использование *референсных метрик*. Наиболее известными из таких метрик являются ATE и RPE [7]. В ATE метрике корректность алгоритма одометрии оценивается сравнением абсолютного расстояния между вычисленной алгоритмом траекторией и референсными данными. В RPE точность траектории задается отношением изменения за фиксированный интервал времени вычисленной траектории к референсной.

Референсные данные могут быть получены с помощью GNSS/INS, систем захвата движений (MoCap) или высокоточных статичных лидаров. Из-за своей погрешности системы GNSS/INS являются эффективными только на большой траектории. MoCap, наоборот, применим лишь в ограниченном пространстве, а высокоточные статичные 3D-сканеры местности, лидары являются крайне дорогими и не подходят для использования в динамике.

В качестве другого подхода исследователи генерируют искусственные карты из облаков точек [1], где получение референсных данных является тривиальной задачей. В данном случае главной сложностью является генерация самих карт — искусственная среда для справедливости оценки должна быть максимально приближена к реальной.

2.2. Безреференсные метрики

Безреференсные метрики основаны на численном измерении шума и консистентности *карты* — множества облаков точек, расположенных

согласно их позициям в траектории. Преимуществом такого подхода является универсальность в ситуациях, когда использование референсных данных затруднительно.

Базовыми алгоритмами считаются Mean Plane Variance (MPV) и Mean Map Entropy (MME). Далее дадим им формальные определения [2, 3, 6]:

Определение 1 (Энтропия) *Энтропия h для точки q_k вычисляется по формуле:*

$$h(q_k) = \frac{1}{2} ln|2\pi e \sum(v_r(q_k))|_{det}$$

где $\Sigma(v_r(q_k))$ — выборочная ковариация точек в v_r-окрестности точки q_k.

Определение 2 (MME) *Безреференсная метрика, считающая резкость карты, и представляющая собой усредненный результат энтропии по всем точкам:*

$$H(Q) = \frac{1}{Q} \sum_{k=1}^{Q} h(q_k)$$

Определение 3 (MPV) *Безреференсная метрика, выражающая искажения карты через усредненную по всем точкам ошибку плоскостей (квадратичное расстояние от точек до оцененной плоскости):*

$$V(Q) = \frac{1}{Q} \sum_{k=1}^{Q} \lambda_{min}$$

где λ_{min} — наименьшее собственное значение выборочной ковариации точек в v_r-окрестности точки q_k.

В работе [6] статистически показано, что данные метрики обладают слабой корреляцией с ошибкой траектории. В своей статье авторы приводят новую метрику — Mutually Orthogonal Metric (MOM), обладающую лучшей корреляцией с RPE.

Определение 4 (MOM) *Безреференсная метрика, результатом которой является среднее арифметическое значений метрики MPV по каждому из направлений предварительно извлеченного ортогонального базиса.*

Важным шагом данного алгоритма является построение ортогонального базиса — извлечение подмножества взаимно ортогональных плоскостей из карты облаков.

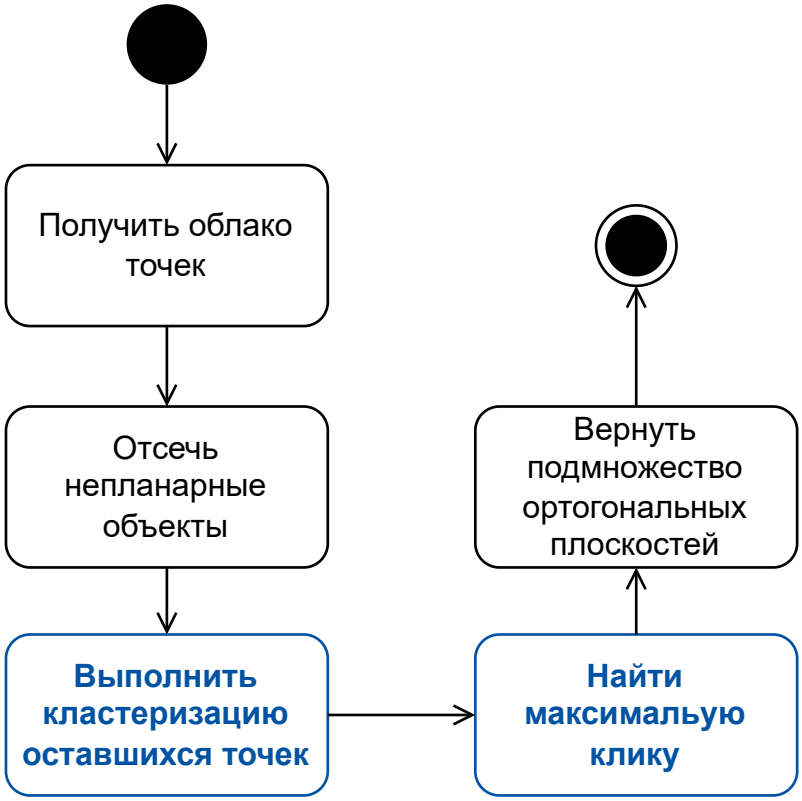

Рис. 1: Диаграмма активностей алгоритма MOM

На представленной диаграмме особый интерес представляют выделенные компоненты. Алгоритмы кластеризации и поиска максимальной кликой являются вычислительноёмкими и нуждаются в грамотном выборе инструментов для реализации.

3. Реализация библиотеки с безреференсными метриками

Текущая реализация[2] библиотеки является *proof-of-concept*. Время работы алгоритмов велико, что, к примеру, ограничивает использование инструмента как функцию потерь в ML/DL пайплайнах. Кроме того, одним из возможных применений библиотеки является использование метрик в режиме реального времени во встраиваемых системах, что недостижимо с имеющейся реализацией. Задачей является подобрать оптимальные инструменты и методы для покрытия наибольшего числа бизнес-сценариев.

3.1. Работа с облаками точек

Потери производительности в текущей реализации в первую очередь вызваны использованием Python в качестве основного языка программирования.

Одним из способов достижения высокой скорости работы алгоритмов метрик является реализация их на компилируемом языке. Наиболее подходящим инструментом для данной задачи является язык программирования C++. Это обусловлено не только известной производительностью данного языка среди языков высокого уровня, но и существованием большого числа библиотек для работы с научными вычислениями. Кроме этого, библиотеку на C++ в дальнейшем можно развить для использования во встраиваемых системах (embedded systems).

Метрики работают с облаками точек, для хранения которых нужны собственные структуры данных. Существует несколько инструментов, предоставляющих классы и методы для работы с облаками точек. Наиболее популярными в этой нише являются PCL [10] и Open3D [12]. Обе библиотеки реализуют похожие структуры и алгоритмы для обработки облаков точек с поддержкой многопоточных вычислений.

Схожесть существующих инструментов усложняет их выбор. От-

[2] https://github.com/MobileRoboticsSkoltech/map-metrics/tree/v0.0.1

дельное внимание стоит уделить производительности библиотеки, которая является критичной в возможных сценариях использования метрик. В частности, в пристальной оценке нуждается реализация такой активно использующейся структуры данных как k-d tree *(k-dimensional tree)*.

Несмотря на обилие полезных функций, PCL обладает неудобным интерфейсом [12] и показывает сравнительно низкую производительность [11].

В Python версии map-metrics используется Python-пакет библиотеки Open3D[3]. Выбор данной библиотеки для C++ нецелесообразен — Open3D обладает достаточно большим количеством зависимостей, что не только увеличивает размер целевого инструмента и время сборки, но и затрудняет дальнейшее его распространение из-за необходимости контролировать наличие всех зависимостей на целевых платформах.

Zampogiannis Konstantinos et al. в [10] предложили новую библиотеку для работы с облаками точек — cilantro[4]. Авторы позиционируют её как легковесный инструмент с минимальным набором зависимостей (Eigen3.3 [5]). Согласно проведенным в данной статье исследованиям, cilantro выигрывает в производительности PCL и Open3D.

3.2. Извлечение ортогональных плоскостей

Из диаграммы 1 в данной секции особый интерес представляют компоненты, ответственные за процессы кластеризации и поиска максимальной клики. Реализация отмеченных алгоритмов невозможна стандартными инструментами языка C++.

Алгоритм кластеризации необходим главным образом для работы с меньшим числом точек в карте, что существенно упрощает затраты при дальнейшем подсчёте максимальной клики. В оригинальной реализации map-metrics используется алгоритм агломеративной иерархической кластеризации со слиянием кластеров посредством метода Уорда.

[3] `https://pypi.org/project/open3d/`

[4] `https://github.com/kzampog/cilantro`

Для максимального приближения к результатам исходной версии рассматривались библиотеки, реализующие конкретно данный алгоритм: alglib[5] и hclust-cpp [8]. Однако возможность задания метода Уорда имеется лишь у библиотеки alglib.

Поиск подмножества ортогональных плоскостей можно свести к NP-полной проблеме поиска максимальной клики в графе. Для работы с графами существует стандартный инструмент — boost::graph[6], реализующий алгоритм Брона-Кербоша поиска всех клик. Благодаря гибкому интерфейсу библиотеки, алгоритм можно уточнить для нахождения максимальной клики.

3.3. Конфигурация под сцены использования

Алгоритмы метрик используют внутри себя множество параметров: максимальное расстояние между соседними точками, количество точек для объединения в кластер, максимальное число соседей у точки, и другие. Необходимость подобной тонкой настройки обусловлена различиями сцен использующихся для получения данных сенсоров. Например, при использовании камер глубины в малых помещениях следует ожидать небольшое расстояние между соседними точкам, тогда как облака из LiDAR-данных на большой местности, как правило, разрежены.

Изначально map-metrics была апробирована лишь на LiDAR-датасете KITTI [4]. Очевидно, что многочисленные сценарии использования библиотеки нуждаются в более тонкой настройке всех параметров. Для этого были созданы несколько конфигураций под типичные сценарии получения данных — использование камер глубины и сенсоров LiDAR. Конфигурации описывают такие параметры как радиус между точками в облаке, максимально допустимое число соседей точки, минимальный размер кластера и другие.

[5] `https://www.alglib.net/dataanalysis/clustering.php`

[6] `https://www.boost.org/doc/libs/1_79_0/libs/graph/`

3.4. Реализация

Для работы с облаками точек была использована библиотека cilantro. В качестве основной библиотеки для работы с матрицами использовалась Eigen. Использующийся в работе стандарт языка программирования — C++17. Для кластеризации была выбрана реализация иерархического алгоритма, представленная в библиотеке alglib. Поиск максимальной клики осуществлен средствами boost::graph.

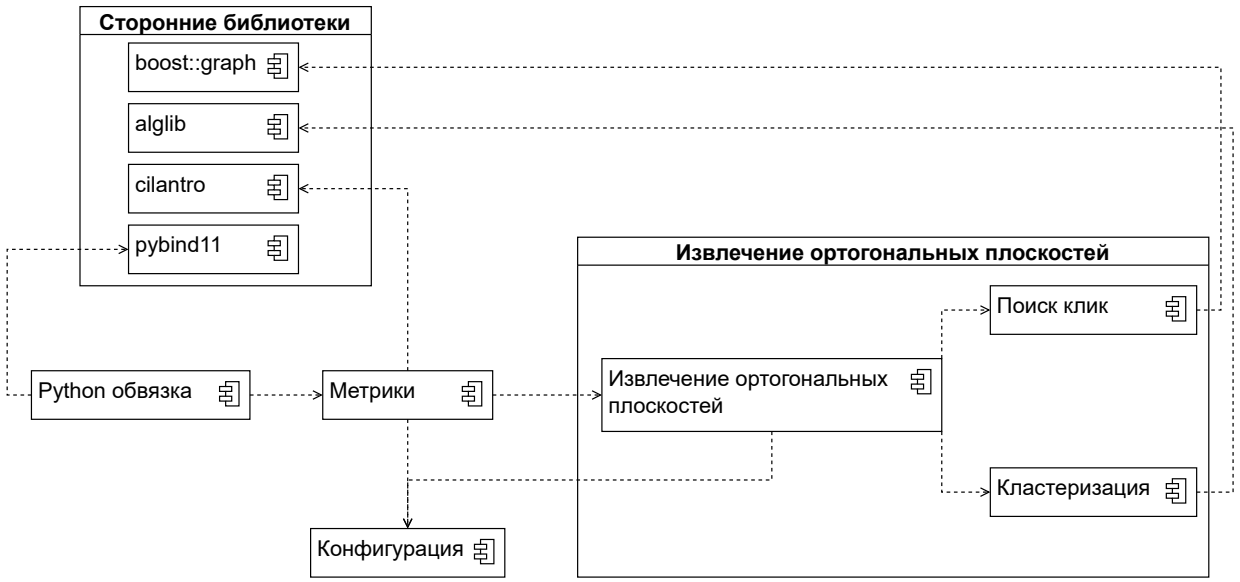

Рис. 2: Диаграмма компонентов реализованной библиотеки

В компоненте «Конфигурация» содержатся классы для задания параметров метрик. Предоставлены два класса конфигурации, покрывающие два наиболее типичных сценария пользования — данные с камер глубины и LiDAR сенсоров. Также существует возможность самостоятельной настройки всех параметров.

Модуль извлечения ортогональных плоскостей, представленный на диаграмме, необходим для работы MOM. Компонента «Python обвязка» отвечает за предоставление Python интерфейса и будет подробно рассмотрена далее.

Исходный код реализованной библиотеки[7] распространяется под свободной лицензией Apache License 2.0.

[7] `https://github.com/MobileRoboticsSkoltech/map-metrics/tree/v0.0.2-alpha`

4. Создание pip-пакета

Язык программирования Python в силу удобного доступа к различным библиотекам пользуется большой популярностью в научном сообществе. Сборка pip-пакета из библиотеки с метриками дает возможность использовать её наряду с другими пакетами научных вычислений. Для создания pip-пакета сперва необходимо предоставить возможность использовать метрики, реализованные на C++, в программе на языке Python — данный процесс именуется *обвязкой*.

4.1. Поддержка Python обвязки

Инструменты для создания Python обвязки для C++ программ разнообразны. Первостепенные отличия выражены в языке, на котором пишется обвязка, и в ограничениях на используемую версию Python и C++.

На текущем этапе был проведён обзор следующих инструментов:

- **Boost.Python**[8] — модуль из семейства библиотек Boost для языка программирования C++. Имеет поддержку numpy типов данных. Boost.Python совместим с большим количеством версий компиляторов C++;

- **Pybind11**[9] — заголовочная библиотека для C++. Имеет интуитивно понятный интерфейс, основанный на интерфейсе Boost.Python. Преимуществом данного инструмента является совместимость с библиотекой Eigen. Pybind11 обязывает пользователя использовать стандарт языка не ниже C++11;

- **Cython**[10] — инструмент, предоставляющий возможность глубокого контроля создания обвязки. Cython имеет собственный язык для написания обвязки, похожий на Python. Данный язык впоследствии транслируется в Си.

[8] https://www.boost.org/doc/libs/1_75_0/libs/python/doc/html/index.html

[9] https://pybind11.readthedocs.io/en/stable

[10] https://cython.readthedocs.io/en/latest/index.html

Большее количество исследуемых критериев и инструментов представлено в следующей таблице:

Таблица 1: Инструменты для создания Python обвязки

Название	Стандарт C++	Версия Python	Поддержка STL	Поддержка Eigen	Лицензия
Boost.Python	C++98	2.2	+	−	Boost Software License
Pybind11	C++11	3.5	+	+	BSD
Cython	C++98	3.3	+	−	Apache License
SIP[11]	C++11	3.6	+	−	GPL2

Из-за повсеместного использования типов данных Eigen в интерфейсе функций, реализованных в проекте, была выбрана библиотека Pybind11. Стоит отметить, что ограничение на версию стандарта языка в данном случае несущественно, так как map-metrics написана с использованием C++17.

4.2. Создание pip-пакета

Написанная версия библиотека map-metrics должна быть доступна широкому кругу пользователей для быстрого скачивания и установки на популярных платформах.

Для достижения поставленной задачи было решено собрать из реализованной библиотеки pip-пакет и сделать его доступным на PyPI[12], индексе пакетов Python.

Существует два формата распространения: **Source Distribution (sdist)**, где вместе с исходными файлами поставляются необходимые метаданные для сборки содержимого пакета на платформе пользователя, и **Wheel (колесо)**, где данные поставляются предсобранными под конкретные платформы. В случае sdist pip-пакет обладает сравнительно большим весом и несёт в себе риски возникновения ошибок во время сборки на платформе пользователя (требуется наличие всех использующихся при сборке зависимостей). Для облегчения процесса установки map-metrics было решено использовать формат распространения в виде колёс.

[11] `https://www.riverbankcomputing.com/software/sip`

[12] `https://pypi.org`

Для семейства операционных систем **GNU/Linux** было решено использовать докер образ manylinux, так как в нём базовым является образ дистрибутива Centos 6 (содержащий достаточно раннюю версию библиотеки glibc, необходимую для сборки). Выбор данной целевой системы обеспечивает совместимость со многими другими дистрибутивами Linux.

Для **macOS** было принято решение осуществлять сборку на macOS 10.15 и поддерживать версии, начиная с OS X 10.14. Минимально поддерживаемая версия 10.14 продиктована ограничениями используемого при реализации библиотеки стандарта C++17.

В случае **Windows**, сборка осуществляется на Windows Server 2019, на которой предустановлен Microsoft Visual Studio Enterprise 2019. Эта версия подходит для обеспечения кроссплатформенности среди наиболее популярных версий Windows: 7, 8, 8.1, и 10.

Автоматическая сборка и публикация pip-пакета map-metrics была настроена с помощью GitHub Actions[13]. Был переиспользован результат, полученный в проекте mrob[14] лаборатории Mobile Robotics Lab. Собранный pip-пакет находится в индексе PyPI[15].

[13] https://github.com/MobileRoboticsSkoltech/map-metrics/actions

[14] https://github.com/MobileRoboticsSkoltech/mrob

[15] https://pypi.org/project/map-metrics/

5. Анализ производительности

Новая версия библиотеки на языке C++ нуждается в численном сравнении с изначальной Python версией, как с точки зрения производительности, так и точности основных алгоритмов, метрик.

5.1. Постановка эксперимента

Главной задачей является анализ времени работы метрик MME, MPV, MOM в базовой версии библиотеки на языке Python и её новой реализации. Исследование призвано показать преимущества версии на C++ и обосновать возможные различия в точности вычислений.

Все эксперименты были осуществлены на предоставленном реальном датасете, который включает в себя позиции траектории и облака точек, полученные с помощью камер глубины. Датасет состоит из 28 облаков (103975 точек), предварительно подвергнутых субдискретизации для обхода физических ограничений тестирующей платформы.

Для результатов, представленных далее в работе, были зафиксированы следующие характеристики среды выполнения: операционная система Linux Ubuntu 18.04.5 64-bit, интерпретатор Python 3.7.12. Аппаратная конфигурация: Intel(R) Xeon(R) CPU @ 2.20GHz, 12GB RAM.

5.2. Время работы алгоритмов

Основной целью является сравнение времени работы алгоритмов в изначальной версии, использующей Open3D, и в версии, реализованной на языке C++ с использованием библиотеки cilantro.

Алгоритмы MME и MPV были запущены на данных 80 раз. Замер времени проводился с помощью метода из Python библиотеки time — process_time_ns, выдающий время в наносекундах, затраченное на выполнение части программы. Результаты приведены в следующей таблице, время дано в секундах:

Таблица 2: Производительность метрик MPV и MME (с)

Метрика	Версия	MIN	MAX	Mean	25%	50%	75%
MME	Py	19.01	19.94	19.27	19.16	19.25	19.37
	C++	**5.12**	**5.48**	**5.27**	**5.21**	**5.27**	**5.33**
MPV	Py	24.06	26.45	24.48	24.31	24.46	24.54
	C++	**5.15**	**6.09**	**5.25**	**5.21**	**5.23**	**5.25**

Из результатов Таблицы 2 видно сильное преимущество новой версии библиотеки. Это обусловлено использованием языка C++ и более быстрой библиотеки для работы с облаками точек, cilantro.

Отметим, что дальнейшее сравнение скорости работы MOM не имеет смысла, так как основные шаги метрики основаны на применении MPV. Однако важным этапом, несвязанным напрямую с самим алгоритмом MOM, является получение подмножества взаимно ортогональных плоскостей.

Таблица 3: Производительность алгоритма извлечения плоскостей (мс)

Версия	MIN	MAX	Mean	25%	50%	75%
Py	349.1	407.9	367.7	359.5	377.0	375.3
C++	**36.3**	**43.7**	**38.4**	**37.2**	**37.8**	**39.5**

Разница в результатах Таблицы 3 обусловлена не только выбором языка программирования, но и небольшими отличиями в реализации алгоритма кластеризации. Корректность подобных изменений численно продемонстрирована в следующем разделе.

5.3. Корректность алгоритмов

Необходимо показать, что в процессе реализации алгоритмов на языке C++ не было допущено ошибок, серьёзно искажающих результаты работы алгоритмов написанных на Python.

Ниже представлены абсолютные разности значений метрик Python и C++ версий:

Таблица 4: Абсолютная разность значений метрик

Δ_{MME}	Δ_{MPV}	Δ_{MOM}
$6 * 10^{-14}$	10^{-18}	$2 * 10^{-18}$

Абсолютная погрешность представленная в Таблице 4 является не более чем ошибкой при работе с числами с плавающей точкой. Можно считать, что точность была соблюдена.

Наибольший интерес вызывает алгоритм извлечения ортогональных плоскостей. Из-за небольших отличий в доступных параметрах для кластеризации в Python и C++ версиях результат работы MOM отличается при использовании ортогонального подмножества, полученного старой и текущей реализацией соответственно.

Однако в данном случае важен не столько абсолютный результат метрики, сколько динамика изменения значений метрики на C++ плоскостях относительно плоскостей, полученных Python алгоритмом.

Для изучения динамики значений алгоритма MOM был взят датасет, состоящий из 4722 облаков. Метрика была запущена на 47 фрагментах карты (по 100 облаков) сперва используя набор ортогональных плоскостей Python алгоритма, затем версии C++.

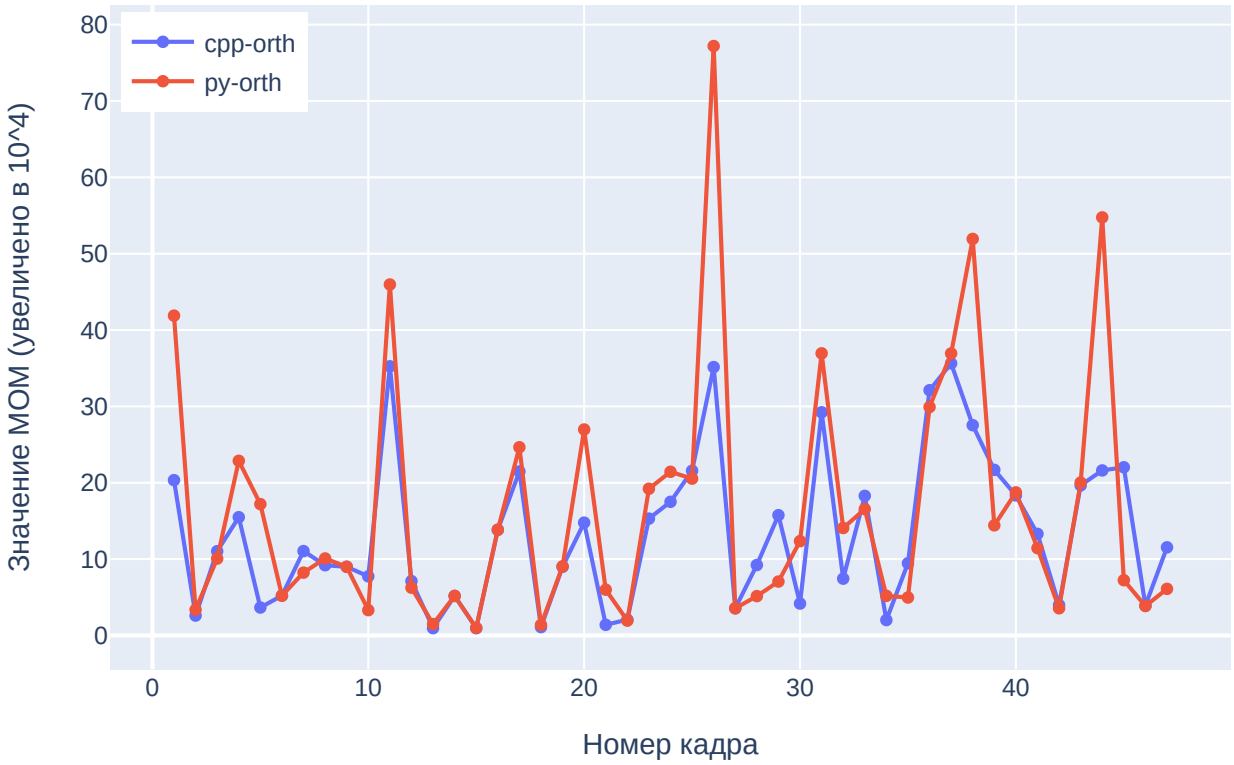

Рис. 3: Поведение МОМ на разных наборах плоскостей

Из графика 3 заметим, что значения МОМ на большинстве кадров обладают схожей динамикой. Кроме этого, было визуально подтверждено (рис. 4) допустимое качество работы реализованного алгоритма извлечения плоскостей.

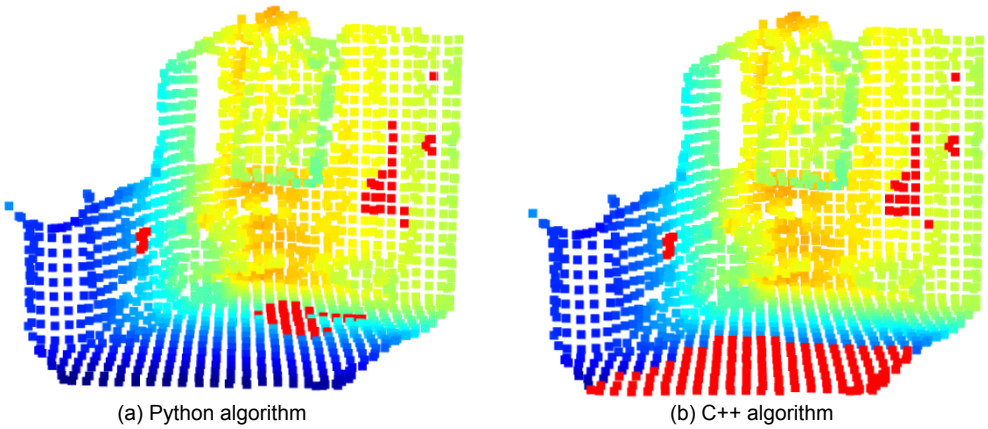

Рис. 4: Ортогональные плоскости (выделены красным)

Заключение

В ходе данной работы были достигнуты следующие результаты:

- проведён обзор существующих методов для оценки алгоритмов одометрии;
- реализована библиотека map-metrics на языке C++;
- добавлена возможность задавать параметры алгоритмов в зависимости от сценария использования датчиков;
- поддержана Python обвязка и CI/CD библиотеки в PyPI;
- проведена оценка качества и времени работы метрик.

Список литературы

[1] CARLA: An Open Urban Driving Simulator / Alexey Dosovitskiy, German Ros, Felipe Codevilla et al. // Proceedings of the 1st Annual Conference on Robot Learning. — 2017. — P. 1–16.

[2] Droeschel D., Stueckler J., Behnke S. Local multi-resolution representation for 6D motion estimation and mapping with a continuously rotating 3D laser scanner // Proc. of the IEEE Int. Conf. on Robotics and Automation (ICRA). — 2014. — May. — P. 5221–5226. — Access mode: `http://ais.uni-bonn.de/papers/ICRA_2014_Droeschel.pdf`.

[3] Evaluation of Registration Methods for Sparse 3D Laser Scans / Jan Razlaw, David Droeschel, Dirk Holz, Sven Behnke. — 2015. — 09.

[4] Geiger Andreas, Lenz Philip, Urtasun Raquel. Are we ready for autonomous driving? The KITTI vision benchmark suite // 2012 IEEE Conference on Computer Vision and Pattern Recognition. — 2012. — P. 3354–3361.

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

[6] Kornilova Anastasiia, Ferrer Gonzalo. Be your own Benchmark: No-Reference Trajectory Metric on Registered Point Clouds // CoRR. — 2021. — Vol. abs/2106.11351. — arXiv : 2106.11351.

[7] Prokhorov David, Zhukov Dmitry, Barinova Olga et al. Measuring robustness of Visual SLAM. — 2019. — 1910.04755.

[8] Müllner Daniel. Fastcluster: Fast Hierarchical, Agglomerative Clustering Routines for R and Python // Journal of Statistical Software. — 2013. — 05. — Vol. 53. — P. 1–18.

[9] Past, Present, and Future of Simultaneous Localization and Mapping: Toward the Robust-Perception Age / Cesar Cadena, Luca Carlone,

Henry Carrillo et al. // IEEE Transactions on Robotics. — 2016. — Vol. 32, no. 6. — P. 1309–1332.

[10] Rusu Radu Bogdan, Cousins Steve. 3D is here: Point Cloud Library (PCL) // 2011 IEEE International Conference on Robotics and Automation. — 2011. — P. 1–4.

[11] Zampogiannis Konstantinos, Fermuller Cornelia, Aloimonos Yiannis. cilantro // Proceedings of the 26th ACM international conference on Multimedia. — 2018. — Oct. — Access mode: http://dx.doi.org/10.1145/3240508.3243655.

[12] Zhou Qian-Yi, Park Jaesik, Koltun Vladlen. Open3D: A Modern Library for 3D Data Processing. — 2018. — 1801.09847.

[13] A benchmark for the evaluation of RGB-D SLAM systems / Jürgen Sturm, Nikolas Engelhard, Felix Endres et al. // 2012 IEEE/RSJ International Conference on Intelligent Robots and Systems. — 2012. — P. 573–580.