

Санкт-Петербургский государственный университет

Вивдик Софья Андреевна

Выпускная квалификационная работа

Разработка алгоритма сегментации
облаков LiDAR на основе сегментации
изображений

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:
младший научный сотрудник, Институт искусственного интеллекта С. Галочкин

Санкт-Петербург
2024

Saint Petersburg State University

Sofia Vivdich

Bachelor's Thesis

LiDAR point cloud segmentation based on image segmentation

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:
Research Scientist, AIRI S. Galochkin

Saint Petersburg
2024

Оглавление

Введение	4
1. Постановка цели и задач	6
2. Обзор	7
2.1. Алгоритмы сегментации облаков точек	7
2.2. Метрики оценки точности сегментации	12
2.3. Наборы данных	15
2.4. Вывод	16
3. Алгоритм сегментации облаков	17
3.1. Принцип работы алгоритма	17
3.2. Используемые инструменты	18
3.3. Подготовка облака к сегментации	19
3.4. Кластеризация точек	27
4. Экспериментальное исследование	33
4.1. Цель и вопросы эксперимента	33
4.2. Условия эксперимента	33
4.3. Результаты	34
4.4. Вывод	39
Заключение	40
Список литературы	41

Введение

В современном мире широкое распространение получили автономные транспортные средства, управление которыми осуществляется без водителя. Такие автомобили планируют движение путем сканирования пространства различными датчиками, анализируя полученную информацию. Популярным компонентом подобной системы датчиков является лидар. Лидар направляет лазерный луч на объект и при отражении получает его обратно, за счет чего вычисляется расстояние до точки. Таким образом, лидар формирует 3D-представление окружающего пространства в виде облака точек. Время суток и погодные условия не оказывают губительное влияние на качество лидарных данных, чего нельзя сказать о снимках камеры. Солнце и уличное освещение создают блики, пасмурная погода и темное время суток делают объекты на фото неразличимыми для механизмов распознавания. В связи с вышеизложенным использование лидара необходимо для получения стабильных данных в любых условиях и, как следствие, для надежной беспилотной навигации.

Для ориентирования автономного средства в пространстве первоочередной задачей является обнаружение и семантическая сегментация объектов в лидарных облаках, то есть разбиение точек на категории в соответствии с их смыслом. Лидарные данные представляют собой десятки тысяч разреженных точек с неравномерной плотностью, поэтому детерминированные алгоритмы не справляются с их обработкой. Для сегментации данных используют нейронные сети, которым для генерации качественных предсказаний необходимы тренировочные данные в виде больших датасетов аннотированных облаков. Ручная разметка требует много средств и времени и не способна охватить все разнообразие географических регионов, множество редких классов объектов, таких как детские коляски и неизвестные животные [23]. Так, в рамках работы [20] была проведена семантическая сегментация облаков одного из самых крупномасштабных наборов данных KITTI [8]. Авторы отмечают, что разметка проводилась не для каждого отдельного разреженного

облака, а сразу для нескольких облаков, объединенных в одно плотное. Несмотря на эту оптимизацию, разметка всего набора данных заняла более 1700 часов. В связи с этим возникает необходимость автоматизировать процесс сегментации лидарных облаков для генерации предсказаний объектов на *плотном облаке*. Это позволит свести процесс ручной сегментации необработанных данных к корректировкам уже готовых предсказаний, что существенно облегчает и снижает затраты на ручную разметку и, как следствие, упрощает генерацию обучающих данных для нейронных сетей. Таким образом, задача сводится к получению предварительной автоматической разметки отдельных объектов в *плотных облаках*.

Беспилотные средства в совокупности с лидарами оснащены камерами. Изображения с них могут быть сегментированы с помощью нейронных сетей, например SAM [18], SEEM [19]. Генерируемая ими разметка довольно точна, так как нейронные сети имеют большие коллекции тренировочных снимков с истинной семантической сегментацией. Так как лидарные облака и снимки камеры отражают одно и то же пространство, но в разных форматах, сегментация изображений с камер может быть переиспользована для облаков. Некоторые современные алгоритмы задействуют разметку объектов со снимков вместе с лидарными облаками. Так, например, работы [9, 10] переносят разметку со снимков на отдельные лидарные облака. Другие методы [6, 23, 26] направлены на изучение дескрипторов для отдельных лидарных облаков, которые содержат полезную для распознавания объектов информацию об изображениях. Однако, оба класса подходов не ориентированы на использование данной разметки для *плотного облака*, с которым приходится работать разметчикам. Более того, в таком сценарии сегментации доступна информация об изображениях со всей протяженности отснятой траектории, то есть сегментация может быть уточнена за счет объединения предсказаний объекта с различных ракурсов съемки.

В данной работе предлагается алгоритм разметки объектов в плотных облаках точек с использованием информации с изображений и современных подходов для распознавания объектов на них.

1. Постановка цели и задач

Целью данной работы является разработка алгоритма сегментации облаков точек на основе разметки 2D-снимков.

Для достижения цели были поставлены следующие задачи.

1. Провести обзор существующих решений сегментации облаков точек, метрик для оценки качества разметки, наборов данных для проведения экспериментов.
2. Предложить и реализовать алгоритм разметки облаков точек на основе сегментации изображений.
3. Определить оптимальные значения гиперпараметров алгоритма.
4. Провести экспериментальное сравнение разработанного алгоритма с ручной разметкой и существующими алгоритмами кластеризации.

2. Обзор

В данном разделе рассматриваются существующие алгоритмы кластеризации и сегментации облаков точек, метрики для оценки качества разметки, а также наборы данных для проведения экспериментов.

2.1. Алгоритмы сегментации облаков точек

Для сегментации облаков часто используются методы на основе нейронных сетей, однако для обучения им необходимы огромные датасеты размеченных облаков, которые в необходимом объеме охватывают разнообразные классы объектов городской среды. В данном разделе представлен обзор существующих решений сегментации облаков и формирования датасетов для дальнейшего обучения нейронных сетей.

2.1.1. Ручная разметка

Один из подходов заключается в ручной обработке и разметке облаков. Процесс такой разметки долгий и дорогой, так как требует времени и средств не только на работу, но и на серьезное обучение специалистов. Охватить ручной разметкой всевозможные географические регионы, временные изменения, разнообразие флоры и фауны, меняющиеся погодные условия невозможно. Кроме того, меняется аппаратное обеспечение, датчики и их настройки, что затрудняет и ограничивает масштабируемость сегментации облаков [30]. Некоторые работы предлагают обучение со слабым контролем [11, 22], однако все еще имеют проблемы с масштабируемостью [17].

2.1.2. Разметка на основе изображений

Подход заключается в сегментации снимков с последующим использованием результатов для разметки облаков точек.

Работа [26] посвящена семантической сегментации облаков на основе 2D и 3D-признаков точек и направлена на формирование правила,

которым можно руководствоваться при выборе оптимального вида, когда точка облака попадает сразу в несколько снимков. Вокруг точки облака формируется прямоугольник заранее заданного размера. Оптимальным считается снимок, на котором видна проекция точки и прямоугольника целиком, у которого площадь прямоугольника наибольшая среди всех остальных подходящих изображений. Далее для каждой точки облака формируется вектор, который содержит 2D (из оптимального снимка) и 3D-признаки. Векторы кластеризуются с помощью алгоритмов Random Forest (построение дерева на основе признаков и объединения точек, листья которого представляют собой семантические метки) и AdaBoost (формирует сильного классификатора итеративным добавлением слабых).

В работе [10] представлена архитектура 2D3DNet, которая использует семантическую разметку 2D-снимков для обучения семантической сегментации 3D-объектов. Этот метод состоит из трех этапов. В первую очередь, с помощью обученной модели для поступивших на вход алгоритма изображений прогнозируется сегментация. На следующем этапе разметки с разных точек обзора объединяются в одну и проецируются в 3D-облако, где определяются семантические метки точек на основе меток отображенных пикселей. Завершающий шаг алгоритма состоит в конвертации меток в 3D-объекты и передаче их сверточной сети для прогноза семантической разметки для каждой точки.

Механизм DriveSegment [6] нацелен на семантическую сегментацию городских сцен. Алгоритм использует лидарные облака и снимки камеры и состоит из следующих этапов.

1. Облако сегментируется, полученные экземпляры проецируются на снимок.
2. Сегменты вырезаются из снимков и передаются в экстрактор признаков, для кластеризации признаков применяется алгоритм k-средних. Каждому сегменту присваивается уникальный номер метки.
3. Модель-учитель итеративно обучается строить прогнозы сегмен-

тации на основе исходных псевдометок. Сегментам присваивается наиболее часто прогнозируемая моделью метка. На полученном наборе меток обучается модель–ученик, которая достигает лучших результатов.

Одна из работ использовала рассматриваемый подход для построения BIM–моделей (Building Information Modelling) в архитектуре для отражения состояния зданий с целью упрощения их обслуживания [12]. Представленный метод заключается в сегментации изображений с помощью архитектуры CNN [29] с последующим переносом результатов сегментации на облако точек. Неточности разметки корректируются на основе геометрической информации об объектах на этапах постобработки, один из которых заключается в добавлении неразмеченных точек и их соседей к границам экземпляров, при условии, что точки удовлетворяют пороговым значениям. Пороговые значения могут быть определены автоматически, однако авторы отмечают, что вычисленные специалистом параметры приводят к более стабильному результату разметки, что является серьезным недостатком метода. Похожая работа была выполнена для построения моделей культурного наследия. Подход, представленный в работе [27], также основан на семантической сегментации 2D–представления облака нейронной сетью CNN [29] с последующим проецированием данных и построением облака в 3D. Метод не предусматривает ситуацию, когда возникает сразу несколько предсказаний сегментации, что регулярно происходит при работе с данными с беспилотного транспорта. Данные обеих работ для проведения дополнительных испытаний не предоставлены, поэтому не могут быть проверены и применимы к разметке дорог и их компонентов.

В работе [9] представлена SLidR — модель самостоятельного обучения разметке облаков точек на основе суперпикселей. Суперпиксель — это объединение визуально похожих по некоторому общему признаку пикселей, которые могут принадлежать одному объекту. В данной работе предлагается сегментация изображения на суперпиксели с помощью метода SLIC [14], после чего к каждой точке облака применяется отображение, переводящее точку в соответствующий пиксель. Считает-

ся, что исходные точки облака, которые попали в один суперпиксель при отображении, принадлежат одному 3D-объекту по точечному признаку, схожему с суперпиксельным признаком.

Метод Seal, представленный в работе [17], направлен на самостоятельное обучение разметке облаков на основе 2D-снимков. Авторы переносят разметку с изображений на облако и отражают их в общее D-мерное пространство, где выполняют группировку экземпляров. Эти действия выполняются для двух пар:

1. снимок и облако в момент времени t ;
2. снимок в момент t и облако в момент $t+n$.

Результат первой пары применяется для пространственного обучения и комбинируется с результатом второй для обеспечения временной согласованности.

Механизм UnScene3D [13] используется для сегментации облаков помещений. Основа алгоритма — генерация предложений потенциальных границ объектов с помощью обученных self-supervised функций. Полученные для снимков и облаков псевдомаски объединяются в общую сцену, после чего сегментация уточняется с помощью итеративного самообучения системы на собственных прогнозах. На каждой итерации с использованием алгоритма [21] передний и задний план сегмента разделяются, после чего выполняется повторная разметка, которая в объединении с исходной будет использована как прогноз на следующей итерации обучения.

2.1.3. Работа только с облаками

Рассмотренные далее механизмы кластеризации и сегментации облаков используют только 3D-знания и работают непосредственно с лидарными данными.

HDBSCAN [4], один из популярных алгоритмов кластеризации, основан на иерархической плотности экземпляров. Плотность региона вокруг выбранной точки вычисляется с помощью расстояния до k -ого соседа — чем меньше расстояние, тем плотнее область. Регионы высокой

плотности представляют собой отдельные кластеры, причем алгоритм способен замечать ситуации, когда необходимо выделить подкластеры для более точных результатов.

Авторы метода [1] вместо сканирования трехмерного облака проводят сегментацию его 2D–изображения для ускорения разметки. Для неразмеченных пикселей выполняется поиск в ширину с целью определить весь рассматриваемый компонент. Решение о добавлении новой точки в очередь BFS принимается на основе геометрических признаков текущей и соседних точек. Алгоритм применяется для разметки облаков *в реальном времени*.

В работе [24] точки трехмерного облака проецируются в двумерную систему координат, в которой и происходит сегментация. Алгоритм преимущественно опирается на геометрические признаки точек. Работа заточена на применение к *разреженным* облакам и экономии вычислительных ресурсов.

В работе [5] представлена пространственная единица в трехмерных сферических координатах — изогнутый воксель. Предложенный на его основе алгоритм кластеризации состоит из четырех этапов:

1. преобразование декартовых координат в сферические;
2. формирование хеш–таблицы, которая хранит пары <индекс вокселя; индексы точек внутри вокселя>;
3. поиск точек в соседних изогнутых вокселях для каждой точки облака;
4. объединение точки и ее соседей в один кластер.

Описанный метод отличается качественной кластеризацией *разреженных* облаков и высокой производительностью за счет работы с хеш–таблицей.

В работе [23] предложен следующий алгоритм кластеризации облаков. В первую очередь отбрасываются точки дороги, для остальных точек определяются начальные предложения сегментации. Вокруг экземпляра выбирается так называемая область интереса, для каждой точки

которой строятся дескрипторы с помощью нейронной сети SegContrast [16]. На основе векторов вычисляется и формируется карта значимости текущего предложения, которая позволяет определить наиболее связанные с экземпляром точки. Следующим шагом строится граф: точки наивысшей значимости и их ближайшие соседи определяются как передний план, остальные — как фон. Завершается процедура применением GraphCut [2] для отделения экземпляра от фона.

Фреймворк, представленный в работе [7], предлагает предварительное сегментирование облака с последующим разделением обработки для ранее известных и неизвестных объектов. Из размеченного облака удаляются известные экземпляры и фон, к оставшимся неизвестным объектам применяется метод эллипсоидальной кластеризации с целью генерации новых для модели семантик. Для точки по некоторым правилам строится эллипсоид соседей. Считается, что точки эллипсоида принадлежат тому же кластеру, что и его центральная точка, кроме того для них также строится эллипсоид для дальнейшего расширения (это не касается точек, которые находятся на расстоянии от центральной точки эллипсоида меньше, чем длина наименьшей оси). Если одной точке соответствуют разные экземпляры, они объединяются и интерпретируются далее как один. Для известных экземпляров проводится процедура избавления от чрезмерной детализации. В основе этого алгоритма поиск близких точек, принадлежащих разным объектам, с последующим их объединением в один экземпляр. Завершается обработка комбинированием всех размеченных объектов.

2.2. Метрики оценки точности сегментации

В данном разделе представлены популярные метрики оценки точности предсказаний сегментации облаков точек.

2.2.1. Pixel accuracy

Метрика Pixel accuracy используется для оценки точности сегментации изображений, но также может применяться для проверки качества

разметки облаков.

Для вычисления оценки каждая точка облака, для которого построено предсказание сегментации, сравнивается с соответствующей точкой в истинной разметке. За каждое совпадение прогноза с истинной сегментацией начисляется один балл, после чего полученное значение усредняется по количеству точек:

$$PixelAccuracy = \frac{1}{n} \sum_{i=1}^n (A_i == B_i, 1, 0)$$

2.2.2. IoU

Intersection over Union — популярная метрика для измерения точности сегментации объектов. Она вычисляет степень совпадения ограничивающих рамок истинного объекта и прогнозируемого.

Пусть A — истинная ограничивающая рамка объекта, B — рамка прогнозируемого алгоритмом объекта. Тогда метрика точности сегментации вычисляется по формуле:

$$IoU = \frac{|A \cap B|}{|A \cup B|}$$

Чем выше значение оценки, тем больше прогноз совпадает с истинным объектом и выше точность сегментации.

2.2.3. mIoU

Метрика mean Intersection over Union отражает значение оценки IoU, усредненное по всем классам распознаваемых объектов.

2.2.4. Precision

Для каждой точки облака в контексте рассматриваемой задачи возможно четыре результата предсказания.

- True Positive (TP) — точка принадлежит физическому объекту и правильно классифицируется алгоритмом как часть этого объекта.

- True Negative (TN) — точка не является частью физического объекта и правильно классифицируется алгоритмом как не относящаяся к объекту.
- False Positive (FP) — точка не является частью физического объекта и ложно классифицируется алгоритмом как часть этого объекта.
- False Negative (FN) — точка принадлежит физическому объекту и ложно классифицируется алгоритмом как не относящаяся к объекту.

С использованием введенных обозначений формула для вычисления метрики *precision* выглядит следующим образом:

$$precision = \frac{TP}{TP + FP}$$

Таким образом, метрика *precision* показывает отношение правильно распознанной части объекта ко всему предсказанию объекта. Иными словами, эта метрика отражает точность разметки — какая часть предсказания в действительности относится к объекту.

2.2.5. Recall

Вычисление метрики *recall* с использованием введенных в пункте 2.2.4 обозначений описывается следующей формулой:

$$recall = \frac{TP}{TP + FN}$$

Таким образом, метрика *recall* показывает отношение правильно распознанной части объекта ко всему реальному объекту. Иными словами, эта метрика отражает полноту разметки — какая доля реального объекта была распознана алгоритмом.

2.2.6. FScore

Метрика FScore учитывает и точность, и полноту предсказаний алгоритма и может быть вычислена по следующей формуле:

$$F\lambda Score = \frac{(1 + \lambda^2) \cdot precision \cdot recall}{(\lambda^2 \cdot precision) + recall}$$

Параметр λ задает приоритеты precision и recall. Как правило в расчетах используют значение $\lambda = 1$, что соответствует равнозначности этих метрик. Тогда формула принимает следующий вид:

$$F1Score = \frac{2 \cdot precision \cdot recall}{precision + recall}$$

Таким образом, метрика FScore позволяет получить более сбалансированную оценку предсказаний алгоритма, чем precision и recall по отдельности.

2.3. Наборы данных

В данном разделе представлены популярные наборы данных, снятых в городской среде, которые могут быть использованы для обучения моделей и тестирования алгоритмов сегментации.

2.3.1. KITTI

KITTI Vision Benchmark Visual Odometry¹ [8] — набор реальных данных, полученных при движении в городской среде. Окружающее пространство снимается несколькими цветными камерами и лидаром, точность локализации обеспечивается использованием GPS. Датасет предоставляет 22 последовательности лидарных облаков. Отличительной особенностью этого датасета является поставка необработанных данных в том виде, в котором они были зафиксированы при движении. Это делает тестирование алгоритмов максимально приближенным к реальным условиям.

¹https://www.cvlibs.net/datasets/kitti/eval_odometry.php (дата обращения: 21-04-2024)

2.3.2. nuScenes

Датасет nuScenes² [28] — это крупномасштабный набор данных, собранных при движении в городской среде Сингапура и Бостона. В состав датчиков включены несколько цветных камер, лидар, GPS, устройство стабилизации. Набор содержит вручную аннотированные облака, которые могут быть использованы для проведения исследований и оценки точности разметки алгоритмами.

2.3.3. Waymo Open

Waymo Open³ [15] — один из крупнейших и самых разнообразных наборов данных автономного вождения. Система датчиков включает в себя пять лидаров (один лидар среднего радиуса действия (вверху) и четыре лидара ближнего действия (спереди, сбоку слева, сбоку справа и сзади)), цветные камеры, GPS, стабилизатор. Предоставляет данные трех категорий: обучающие, тестовые и проверочные. Особое внимание уделено обучающим данным для развития предсказаний разметки.

2.4. Вывод

Таким образом, можно заключить, что многие существующие решения сегментации облаков используют только 3D-данные и строят предсказания с помощью нейронных сетей. В то же время, основываясь на проведенном обзоре, можно выделить группу инструментов, дополнительно использующих 2D-знания для улучшения результатов разметки.

Кроме того, существует ряд крупномасштабных наборов реальных данных городской среды, которые активно используются для обучения нейронных сетей, а также в качестве необработанных данных для проведения сегментации облаков новыми алгоритмами, предсказания которых могут быть оценены популярными метриками.

²<https://www.nuscenes.org/> (дата обращения: 21-04-2024)

³<https://waymo.com/open/> (дата обращения: 21-04-2024)

3. Алгоритм сегментации облаков

В данном разделе представлен альтернативный подход сегментации облаков точек. Алгоритм состоит из двух этапов: многоступенчатая обработка исходных лидарных данных с целью повышения характеристик, влияющих на разметку, и кластеризация алгоритмом GraphCut [3, 2].

3.1. Принцип работы алгоритма

Сегментация облаков точек идейно распадается на два этапа, что стало основой предлагаемого решения.

Первый этап состоит в подготовке облака к сегментации. Изначально каждое облако датасета представляет собой необработанный набор точек, отражающий состояние окружающего пространства при конкретном положении машины с оборудованием. Этот определенный момент съемки задается для каждого облака в виде матрицы трансформации. Такие облака довольно разрежены, то есть каждый выбранный участок покрывается небольшим количеством точек, что в дальнейшем может оказывать влияние на качество разметки, особенно в вопросе границ объектов. В связи с этим первый шаг подготовки облака к сегментации заключается в повышении плотности облака путем объединения последовательности облаков в одно. Полученное *плотное* облако покрывается последовательностью снимков камеры, в результате чего каждой 3D-точке сопоставляется 2D-пиксель, маска сегментации которого и считается маской точки. Затем полученные таким образом первичные знания о принадлежности точек облака конкретным объектам на основе разметки снимков проходят череду улучшений:

1. из рассмотрения выбрасываются точки, соответствующие пиксели которых при разметке снимков не попали в маски;
2. выбор близкого к датчикам фрагмента карты;
3. объединение точек в воксели;

4. удаление шума.

Второй этап заключается в сегментации обработанного облака с использованием алгоритма GraphCut [3, 2]. Для работы алгоритма как признак, на основе которого будет выполняться разметка, требуется матрица расстояний между точками подготовленного облака. По матрице расстояний легко обнаружить иногда встречающиеся изолированные точки, которые ощутимо удалены от основной массы. Для корректной работы алгоритма разметки их необходимо удалить из рассмотрения. Следующим шагом вызывается GraphCut [3, 2], результат работы которого — объединенные в кластеры точки облака.

Далее описываются использованные при разработке инструменты и подробно раскрывается реализация каждого шага алгоритма.

3.2. И использованные инструменты

Для реализации алгоритма был выбран язык Python, так как он удобен для работы с 3D-данными через библиотеку Open3D⁴ [25]. Open3D предоставляет набор алгоритмов для оптимизированной параллельной обработки 3D-данных и их визуализации, которые активно использовались в этой работе. Основная функциональность опирается на работу с библиотеками numpy⁵, numpy_indexed⁶, scipy⁷.

Перед началом разработки алгоритма была проведена сегментация 2D-изображений камеры используемого датасета KITTI⁸ [8], что позволило существенно сократить время на разметке картинок при каждом запуске. Снимки были размечены алгоритмом SAM [18], который представлен библиотекой segment_anything⁹. Релиз инструмента состоялся в 2023 году. В работе [18] представлены результаты разметки алгоритмом разнообразных наборов данных, которые были оценены с помощью

⁴<https://pypi.org/project/open3d/> (дата обращения: 21-04-2024)

⁵<https://numpy.org/> (дата обращения: 21-04-2024)

⁶<https://pypi.org/project/numpy-indexed/> (дата обращения: 21-04-2024)

⁷<https://scipy.org/> (дата обращения: 21-04-2024)

⁸https://www.cvlibs.net/datasets/kitti/eval_odometry.php (дата обращения: 21-04-2024)

⁹<https://segment-anything.com/> (дата обращения: 21-04-2024)

метрик IoU и mIoU, рассмотренных в пункте 2.2. Метрики показывают превосходство качества сегментации в сравнении с аналогами и лишь немного уступают ручной разметке. В связи с этим для сегментации снимков в данной работе среди альтернативных инструментов был выбран SAM.

Для сегментации предобработанных облаков в этой работе применяется алгоритм GraphCut [3, 2].

3.3. Подготовка облака к сегментации

В данном разделе описаны этапы обработки облака, выполнение которой задает первичную разметку и выделяет наиболее значимые для сегментации точки.

3.3.1. Формирование карты из последовательности облаков

Как было отмечено выше, исходные лидарные облака довольно разрежены. При сегментации таких облаков можно наблюдать снижение качества разметки на границах объектов. В связи с этим цель первого шага обработки облака заключается в его обогащении точками соседних облаков, снятых примерно в то же время.

Для разметки облака, соответствующего i -ому положению и называемого в дальнейшем «стартовое», рассматривается последовательность облаков в положениях $[i - n, i + m]$, где параметры n, m подбираются экспериментально. На данный момент универсальным решением являются последовательности из четырех–пяти облаков. Для повышения плотности стартового облака необходимо все облака последовательности объединить в одно, предварительно сместив их к положению стартового облака с помощью матричных преобразований.

Полученное *плотное* облако в дальнейшем называется «карта».

3.3.2. Отображение точек облака в маски пикселей

Беспилотные транспортные средства снабжены несколькими типами датчиков, к которым в том числе относятся камеры, которые па-

раллельно с лидарами проводят съемку окружающего пространства. Снимки камеры также соответствуют конкретным положениям. Таким образом, для каждого положения в датасете имеется набор данных, состоящий из облака и снимка, о местности в конкретный момент времени и локации машины. В связи с этим 3D-знания, полученные в результате объединения облаков на шаге 3.3.1, могут быть дополнены 2D-данными за счет покрытия карты последовательностью снимков.

На этом этапе необходимо сформировать двумерный массив предварительной разметки. Строка массива с номером i в столбце j должна содержать номер маски, в которую попал соответствующий пиксель точки i в сегментированном снимке j . Для этого точке карты ставится в соответствие пиксель предварительно сегментированного снимка. Номер маски, в которую попал пиксель при разметке, записывается в матрицу как маска точки. Операция повторяется для каждой точки и каждого снимка.

Небольшой фрагмент полученной матрицы для пяти точек карты и семи снимков выглядит следующим образом:

```
array([[12, 12, 12, 14, 14, 13, 13],  
       [12,  0, 12, 14, 12,  0,  0],  
       [12,  0, 12, 15, 12,  0, 13],  
       [ 0,  0, 10, 15, 12, 11, 13],  
       [ 0, 16, 16, 15, 12, 11, 13]])
```

Так, например, точка №3 на снимке №2 попала в 10-ую маску.

Описанный формат хранения данных о предварительной разметке удобен по нескольким причинам. Во-первых, для большого множества точек формат хранения масок в виде одного двумерного массива целых чисел максимально компактный. Во-вторых, на каждом следующем шаге обработки карты часть точек будет выброшена из рассмотрения и, соответственно, должна быть удалена из матрицы. В текущей реализации матрицы первичных масок эта задача легко решается и сводится к выборке строк по указанным индексам подходящих точек.

3.3.3. Обработка масок SAM

Инструмент сегментации изображений SAM генерирует чрезмерно детальную для рассматриваемой задачи разметку. Так, на рисунке 1 SAM определил каждую составляющую автомобиля как отдельную маску (колесо, дверь, зеркало).



Рис. 1: Разметка снимка алгоритмом SAM

Для того чтобы в итоговой сегментации получить крупные маски, необходимо уже на этапе первичной разметки при переносе масок с изображений на облако объединять мелкие сегменты в более крупные части. Для этого в рамках работы был реализован механизм, который при рассмотрении масок определяет, относятся ли они к одному физическому объекту, и если да, то объединяет их. Для принятия решения о принадлежности масок одному объекту должен быть выполнен хотя бы один из следующих критериев.

1. Площадь пересечения ограничивающих рамок масок составляет 50% и более от площади объединения. Этот критерий позволяет добиться объединения нескольких масок в том случае, когда их размеры примерно одного порядка.
2. Число пикселей в пересечении масок составляет 50% и более от количества пикселей в одной из масок. Этот критерий используется для объединения масок, размеры которых существенно отличаются, в результате чего первый критерий не может быть применен.

Результат работы алгоритма снижения детализации масок SAM представлен на рисунке 2.



Рис. 2: Разметка снимка алгоритмом SAM после снижения детализации

3.3.4. Удаление скрытых точек

При построении массива масок на шаге 3.3.2 для каждого рассматриваемого снимка камеры перед сопоставлением точкам пикселей необходимо из исходной карты удалить скрытые точки. Точка называется скрытой, если она не видна при просмотре облака с указанной точки обзора. При построении матрицы масок точкой обзора считается место крепления камеры, так как цель состоит в удалении из карты точек, которые заведомо не могут быть видны на снимке.

Для реализации такой обработки в первую очередь необходимо перевести карту в систему координат камеры с помощью матричных преобразований. Когда карта находится в системе координат камеры и точка обзора зафиксирована, вызывается `hidden_point_removal`¹⁰ — встроенная функция библиотеки Open3D, которая удаляет скрытые точки.

Описанные действия проводятся при построении матрицы масок для карты относительно каждого снимка последовательности.

3.3.5. Выборка размеченных точек

Алгоритм SAM [18], с помощью которого предварительно была проведена сегментация снимков камеры, не каждый пиксель картинки от-

¹⁰https://www.open3d.org/docs/latest/python_api/open3d.geometry.PointCloud.html#open3d.geometry.PointCloud.hidden_point_removal (дата обращения: 21-04-2024)

носит к какому-то объекту. Существует ряд случаев, когда часть снимка остается неразмеченной (на рисунках 1, 2 такие точки раскрашены в черный). Например, здание белого цвета, расположенное вдоль дороги, может быть не распознано как отдельный объект, если оно освещено солнцем. В матрице масок соответствующим точкам облака устанавливается нулевое значение маски для рассматриваемого снимка.

На данном этапе необходимо оставить для дальнейшей обработки только те точки карты, которые были распознаны как объект хотя бы на одном снимке. С разработанным форматом хранения масок в виде двумерного массива эта задача сводится к выборке ненулевых строк из матрицы. Точки с индексами строк, которые не попали в выборку, удаляются из карты.

Таким образом, дальнейшая обработка проводится с точками, которые имеют высокие шансы оказаться частью некоторого объекта и, как следствие, быть размеченными.

3.3.6. Выборка близких к датчикам точек

Датчики сканируют местность на несколько десятков метров вперед, из-за чего захватывают пространство внушительной площади. Так, объем необработанного лидарного облака в одном положении может составлять сотни тысяч точек. При рассмотрении в рамках разметки всего облака целиком становится невозможным вычислить и хранить матрицу расстояний (пункт 3.4.1) для такого количества точек. Кроме того, далекие объекты слабо видны на камере, стороны дороги сливаются в одну точку из-за перспективы. Все это приводит к размытости удаленных объектов на снимках и, как следствие, к неточностям разметки.

Для повышения производительности и качества разметки было принято решение рассматривать только близкие к датчикам точки, находящиеся внутри куба со стороной около 10 — 20 метров от места установки датчика. Для реализации выборки точек внутри куба использовалась библиотека Numpy. Приведенные экспериментально установленные средние значения стороны достаточны для того, чтобы под дальней-

шее рассмотрение попало несколько объектов, качественно обозримых с датчиков.

3.3.7. Понижение разрешения облака

После всех выполненных к текущему моменту преобразований карта по-прежнему содержит несколько десятков тысяч точек, что довольно объемно для дальнейшей кластеризации. В связи с этим на данном этапе проводится модификация облака, которая заключается в понижении разрешения с помощью объединения точек в воксели. Такой подход позволяет в несколько раз уменьшить количество точек с сохранением общего вида карты и ценной для разметки информации.

Библиотека Open3D для понижения разрешения облака предоставляет следующие функции:

- `voxel_down_sample`¹¹ имеет параметр в виде размера вокселя. Объединяет несколько близких точек в один воксель заданного размера. Координаты вокселя вычисляются путем усреднения координат входящих в него точек. Результатом работы алгоритма является вокселизированное облако, нумерация точек которого не связана с нумерацией в карте.
- `voxel_down_sample_and_trace`¹² аналогично предыдущей функции объединяет точки в воксели. Возвращаемым значением является кортеж, состоящий из вокселизированного облака и списка объединяемых в воксель исходных точек карты.

Так как маски сегментированной после всех преобразований карты будут перенесены на исходное облако, критически важно на каждом этапе иметь соответствие между точками до выполнения очередной обработки и после. В связи с этим использование `voxel_down_sample` не

¹¹https://www.open3d.org/docs/latest/python_api/open3d.geometry.PointCloud.html#open3d.geometry.PointCloud.voxel_down_sample (дата обращения: 21-04-2024)

¹²https://www.open3d.org/docs/latest/python_api/open3d.geometry.PointCloud.html#open3d.geometry.PointCloud.voxel_down_sample_and_trace (дата обращения: 21-04-2024)

подходит ввиду невозможности совершить обратный переход от воксельного облака к исходному. Кроме того, теряется информация о предварительной разметке, из-за чего вокселям нельзя назначить маски.

Таким образом, для понижения разрешения облака была выбрана функция `voxel_down_sample_and_trace`. Карта из трех облаков при вызове функции со значением параметра `voxel_size=0.2` уменьшается с 50 тысяч точек до двух тысяч. Список формирования первых трех вокселей на основе объединения исходных точек выглядит следующим образом:

```
[IntVector[43131, 43132, 43133, 43134],  
 IntVector[43104, 43110, 43112],  
 IntVector[43098, 43100]]
```

На основе этой информации есть возможность, во-первых, после кластеризации вернуться к исходной карте с высоким разрешением, развернув точки из определения каждого вокселя. Во-вторых, приведенный список позволяет построить матрицу масок для воксельного облака с новой нумерацией точек. Для этого была написана функция, которая с помощью библиотеки `numpy_indexed`¹³ выбирает самый часто встречающийся одномерный массив масок среди объединенных точек и устанавливает его как маску вокселя. Так, для вокселя №1 приведенного фрагмента списка, сформированного путем объединения трех точек, массив масок будет самым частым результатом среди массивов масок участвующих в объединении точек.

3.3.8. Удаление шума

На сегментацию облака негативное влияние оказывает шум — группа точек, значительно удаленная от основной массы. Пример такого явления можно увидеть на рисунке 3.

Для удаления шумовых точек в данной работе используется функция `remove_statistical_outlier`¹⁴ библиотеки Open3D. Алгоритм определяет шум как точки, соседи которых находятся дальше, чем среднее

¹³<https://pypi.org/project/numpy-indexed/> (дата обращения: 21-04-2024)

¹⁴https://www.open3d.org/docs/latest/tutorial/Advanced/pointcloud_outlier_removal.html#Statistical-outlier-removal (дата обращения: 21-04-2024)

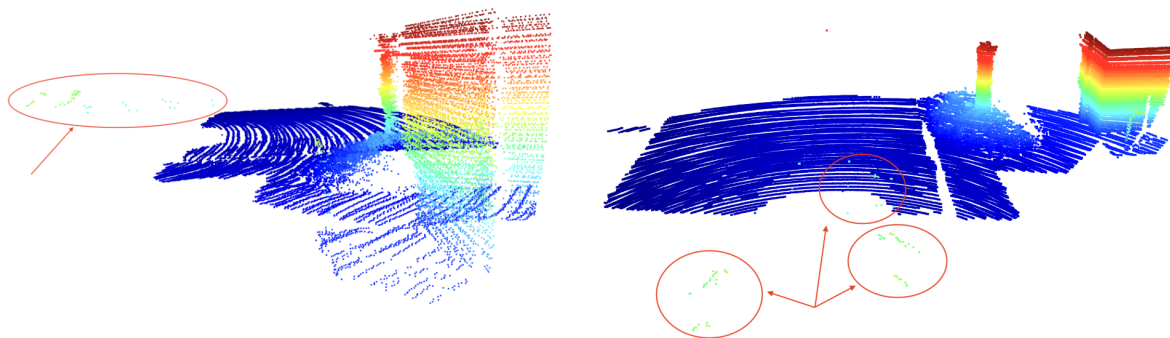


Рис. 3: Шумовые точки относительно основной массы точек с разных ракурсов

расстояние между соседними точками для всего облака. Функция принимает на вход два параметра:

- `nb_neighbors` — количество соседних точек, которые учитываются при вычислении среднего расстояния;
- `std_ratio` — максимально допустимое отклонение от среднего расстояния. Агрессивность функции обратно пропорциональна этому параметру.

Экспериментальные запуски с разными значениями параметров показали, что шумовые точки гарантированно удаляются при маленьких значениях параметра `std_ratio`. Однако вместе с этим также удаляется часть точек в основной массе облака, что снижает плотность и может сказаться на точности сегментации. В связи с этим на данном этапе используется довольно щадящее значение параметра, в 2.5 раза превышающее значение по умолчанию. Такой подход позволяет избавиться от большей части шумовых точек без ущерба основной массе обрабатываемого облака. Оставшаяся часть шума будет распознана и удалена из рассмотрения после вычисления расстояний между точками перед кластеризацией.

3.3.9. Обзор подготовленного облака

В результате проведения рассмотренных подготовительных этапов было получено облако, которое обладает рядом преимуществ по срав-

нению с исходным.

- Высокая плотность за счет объединения нескольких облаков.
- Только значимые для кластеризации точки, которые на снимках были распознаны как физические объекты.
- Точки расположены близко к датчикам, за счет чего максимально четко покрываются 3D– и 2D–данными.
- Минимальное количество шумовых точек.

В разделе 4.3.1 представлено визуальное сравнение первичной разметки исходного облака и карты, обработанной описанными алгоритмами.

3.4. Кластеризация точек

Алгоритмы, приведенные в данной секции, выполняются после завершения всех подготовительных этапов над картой и состоят в формировании признака кластеризации с последующей разметкой.

3.4.1. Вычисление расстояний между точками

Для кластеризации облака точек в первую очередь необходимо выявить признак, который будет отражать вероятность того, что две точки принадлежат одному объекту. В данной работе признаком кластеризации является расстояние между точками, которое определяется степенью сходства точек по номерам их первичных масок и евклидовым расстоянием.

Вычисление расстояния между парой точек по маскам представлено на листинге 1.

Листинг 1: Distance calculation

```
view_counter = 0
for view in range(num_views):
    instance_id1 = instances[point1, view]
    instance_id2 = instances[point2, view]

    if instance_id1 != 0 and instance_id2 != 0:
        view_counter += 1
        if instance_id1 != instance_id2:
            dist[point1, point2] += 1
if view_counter:
    dist[point1, point2] /= view_counter
    dist[point2, point1] = dist[point1, point2]
```

Фиксируются две точки — `point1` и `point2`, их массивы масок поэлементно сравниваются. При ненулевом значении обеих масок конкретного снимка счетчик `view_counter` увеличивается на единицу. Если значения в рассматриваемых ячейках, помимо того что ненулевые, не совпадают, то значение `dist` для соответствующих точек также увеличивается на единицу. Эти действия выполняются для всех элементов массивов масок, после чего расстояние между точками записывается как результат деления значения в `dist` на `view_counter`.

Реализованное таким образом вычисление расстояний при полном совпадении масок у пары точек будет выдавать результат 0, что будет интерпретировано графовым алгоритмом `GraphCut` как отсутствие пути и нулевая вероятность принадлежности точек одному объекту. В то же время точки, маски которых полностью различны, в матрице расстояний `dist` будут иметь значение 1, что наоборот будет восприниматься алгоритмом кластеризации как полное совпадение. В связи с этим значения необходимо «перевернуть».

Итоговое расстояние между парой точек вычисляется следующим образом:

$$label_dist[i, j] = \begin{cases} e^{-\beta \cdot dist[i, j]} \cdot e^{-\alpha \cdot spatial_dist[i, j]}, & spatial_dist[i, j] \leq PT \\ 0, & \text{иначе} \end{cases}$$

В приведенной формуле `spatial_dist` представляет собой евклидово расстояние между точками облака, вычисленное с помощью функции `cdist`¹⁵ библиотеки `scipy`. Параметр PT (Proximity Threshold) отражает максимальное расстояние между точками, которые могут принадлежать одному объекту. В предлагаемом алгоритме используется значение $PT = 3$ метра, так как размеры подавляющего большинства объектов движения и уличного оснащения (лавки, мусорные баки, знаки) не превышают этого значения. Если же длина объекта больше 3 метров, то между парой удаленных точек A и B в рамках этого объекта найдется точка C , расстояние от которой до A и B будет ненулевым (за счет физического расстояния ≤ 3 метра и совпадения масок), что будет учтено алгоритмом `GraphCut`. Оптимальные значения весов β и α и их влияние на результат сегментации подробно исследуются в разделе «Эксперименты» 4.3.2.

Таким образом, физически близкие точки с полностью совпадающими массивами масок, для которых значение `dist` было равным 0, в результирующей матрице расстояний `label_dist` будут иметь значение близкое к 1. Мотивация использования экспоненциальной функции будет описана в разделе 3.4.3, где построенная матрица расстояний `label_dist` будет использована как признак кластеризации.

3.4.2. Удаление изолированных точек

Как было подробно описано в разделе 3.3.8, шумовые точки не всегда полностью удаляются при подготовке карты, что в итоге приводит к появлению нескольких компонент связности в матрице расстояний `label_dist`. Существенная удаленность шумовых точек от остальных проявляется в нулевом расстоянии до них за счет использования параметра PT при построении матрицы в пункте 3.4.1. Такие точки удаляются перед кластеризацией путем выделения наибольшей компоненты связности, так как не представляют интереса в разметке и могут оказывать негативное влияние на результаты работы `GraphCut`.

¹⁵<https://docs.scipy.org/doc/scipy/reference/generated/scipy.spatial.distance.cdist.html#scipy-spatial-distance-cdist> (дата обращения: 21-04-2024)

3.4.3. Алгоритм GraphCut

В данной работе для сегментации используется алгоритм GraphCut [3, 2]. Для его работы точки облака рассматриваются как вершины графа. Весом ребра между парой вершин устанавливается значение из матрицы расстояний `label_dist` для соответствующих точек. Нулевое значение в матрице `label_dist` интерпретируется как отсутствие ребра.

Вершины полученного графа могут быть разбиты на непересекающиеся множества путем удаления ребер, связывающих разделенные части. Алгоритм работает рекурсивно, то есть сначала разделяет вершины на две части, после чего продолжает разбиение каждого фрагмента. Полученные непересекающиеся множества являются искомыми кластерами облака.

Алгоритм GraphCut при кластеризации опирается на минимальный разрез графа. Как было отмечено выше, разбиение на непересекающиеся множества вершин происходит с помощью удаления ребер. Множество удаленных ребер определяет разрез.

Пусть дан взвешенный неориентированный граф $G = \{V, E\}$, где V — множество вершин, E — множество ребер. Разбиение графа состоит в выделении множеств вершин A и B таких, что $A \cap B = \emptyset$, $A \cup B = V$. Степень сходства этих групп вершин вычисляется по формуле:

$$cut(A, B) = \sum_{i \in A, j \in B} W_{ij}$$

где W_{ij} — вес ребра, соединяющего i -ую и j -ую вершины графа. Таким образом, в некоторых реализациях алгоритма степень сходства групп, вычисленная по приведенной формуле, является критерием кластеризации, который необходимо минимизировать для оптимального разбиения графа.

В данной работе для оценки разбиения используется нормализованный критерий сходства групп, предложенный в работе [21]:

$$Ncut(A, B) = \frac{cut(A, B)}{assoc(A, V)} + \frac{cut(B, A)}{assoc(B, V)}$$

$$assoc(A, V) = \sum_{i \in A, k \in V} W_{ik}$$

Помимо альтернативного критерия сходства, работа [21] представляет алгоритм поиска разреза, при котором достигается наименьшее значение $Ncut$. Для этого необходимо построить вспомогательную матрицу на основе весовой матрицы W и вычислить ее собственные векторы, из которых для дальнейших шагов выбирается соответствующий второму наименьшему собственному числу. Далее формируется набор пороговых значений, равномерно распределенных в диапазоне собственного вектора, для каждого из которых вычисляется $Ncut$. Оптимальным разбиением множества на две части считается порог, при котором $Ncut$ принимает наименьшее значение. Для каждой полученной в результате разбиения группы вершин рекурсивно выполняется та же процедура до тех пор, пока значение $Ncut$ очередного разбиения не станет больше заданного параметра T . Универсальное значение параметра для корректной кластеризации большинства облаков подбирается экспериментально и подробно исследуется в разделе «Эксперименты» 4.3.2.

Стоит отметить, что при любом выборе критерия сходства в дальнейшем его значение необходимо минимизировать. Это означает, что различия между точками, выраженные в виде весов ребер, должны быть как можно более явными и значимыми. Именно по этой причине — с целью увеличения разброса между точками — матрица расстояний `label_dist` в пункте 3.4.1 вычислялась через экспоненциальную функцию.

3.4.4. Перенос результатов кластеризации на исходную карту

Результатом разметки облака алгоритмом GraphCut является набор кластеров, каждый из которых представлен множеством точек, которые были ассоциированы с этим кластером. Стоит отметить, что сегментация была выполнена для облака, прошедшего все этапы предобработ-

ки из раздела 3.3. Теперь результаты разметки необходимо перенести на исходную плотную карту из пункта 3.3.1.

Первым этапом раскраски карты является присваивание уникального цвета каждому кластеру. Затем необходимо каждую точку исходной карты ассоциировать с объектом и, как следствие, цветом. Для этого выполняется цикл по кластерам, для каждого из которых выполняются следующие действия.

1. Выборка всех точек подготовленного облака, ассоциированных с обрабатываемым кластером.
2. Каждая точка из предыдущего пункта является вокселем, который был сформирован точками исходной карты. Для этого была использована функция `voxel_down_sample_and_trace` на шаге 3.3.7, одно из возвращаемых значений которой — список объединенных в один воксель точек. Каждой точке из этого списка присваивается цвет обрабатываемого на текущей итерации кластера.

Таким образом, сегментация подготовленного облака переносится на исходную карту.

4. Экспериментальное исследование

В данном разделе описаны цель, исследовательские вопросы и условия экспериментального сравнения, а также продемонстрированы результаты и выводы из них.

4.1. Цель и вопросы эксперимента

Целью настоящего эксперимента является определение качества разметки облаков точек. Для достижения цели были поставлены следующие исследовательские вопросы.

RQ1: Как изменяется карта относительно исходного облака после многоступенчатой обработки перед кластеризацией?

RQ2: Каковы оптимальные значения гиперпараметров алгоритма: веса α и β в функции вычисления расстояния между точками и T как порог завершения рекурсии в GraphCut?

RQ3: Какова точность и полнота сегментации *плотных* облаков точек предложенным алгоритмом относительно существующих решений?

4.2. Условия эксперимента

В данном разделе описывается тестовый стенд, на котором проводилось экспериментальное исследование.

Наборы данных

Эксперименты были проведены на облаках точек и цветных снимках нулевой последовательности набора данных KITTI¹⁶ [8] с достоверными положениями. Последовательность содержит 4540 облаков и изображений. Также датасет содержит истинную сегментацию каждого облака точек, которая включает в себя две составляющие: сегментация объектов движения (транспортные средства, пешеходы и тд) без указания класса и семантическая сегментация, при которой каждой точке присвоена метка класса, к которому относится соответствующий объект. В

¹⁶https://www.cvlibs.net/datasets/kitti/eval_odometry.php (дата обращения: 21-04-2024)

рамках данной работы сегментация объектов движения без указания класса будет считаться эталонной при вычислении метрик. Это обосновано тем, что с точки зрения семантической сегментации различные физические объекты одного и того же класса (например, несколько машин) алгоритмически неразличимы, так как их точки имеют одинаковые метки. Для проведения настоящего эксперимента критически важно проверить способность предложенного алгоритма находить и выделять отдельные физические объекты, поэтому в результате его работы объекты одного класса будут распознаны как независимые компоненты.

Метрики

Для оценки точности сегментации используется метрика *precision*, подробно рассмотренная в 2.2.4. Полнота разметки оценивается метрикой *recall* (пункт 2.2.5). Заключение о качестве сегментации выдается метрикой *FScore* (пункт 2.2.6) с параметром $\lambda = 1$.

4.3. Результаты

В данном разделе представлены результаты экспериментов для ответа на исследовательские вопросы.

4.3.1. Обзор подготовленного к сегментации облака

Эксперимент был проведен на последовательности облаков №20–24 набора данных KITTI. Четыре облака последовательности были объединены в одну карту. В качестве покрывающих снимков был взят набор картинок №20–24.

Точки карты были ассоциированы с пикселями снимков в соответствии с пунктом 3.3.2 предобработки облака. Результат раскраски точек облака в цвета соответствующих масок при разметке снимка №20 представлен на рисунке 4. Стоит отметить, что раскраска точек подсветила ряд проблем у необработанного облака:

- наличие шумовых и скрытых точек;

- захват территории большого радиуса, что приводит к размытости удаленных от датчиков объектов и в целом увеличивает объем карты для дальнейшей обработки;
- наличие точек, которые заведомо не могут быть распознаны как объекты, что подтверждается невозможностью их разметки на уровне сегментации снимка (на рисунке 4 в зоне видимости датчиков такие точки раскрашены в черный цвет).

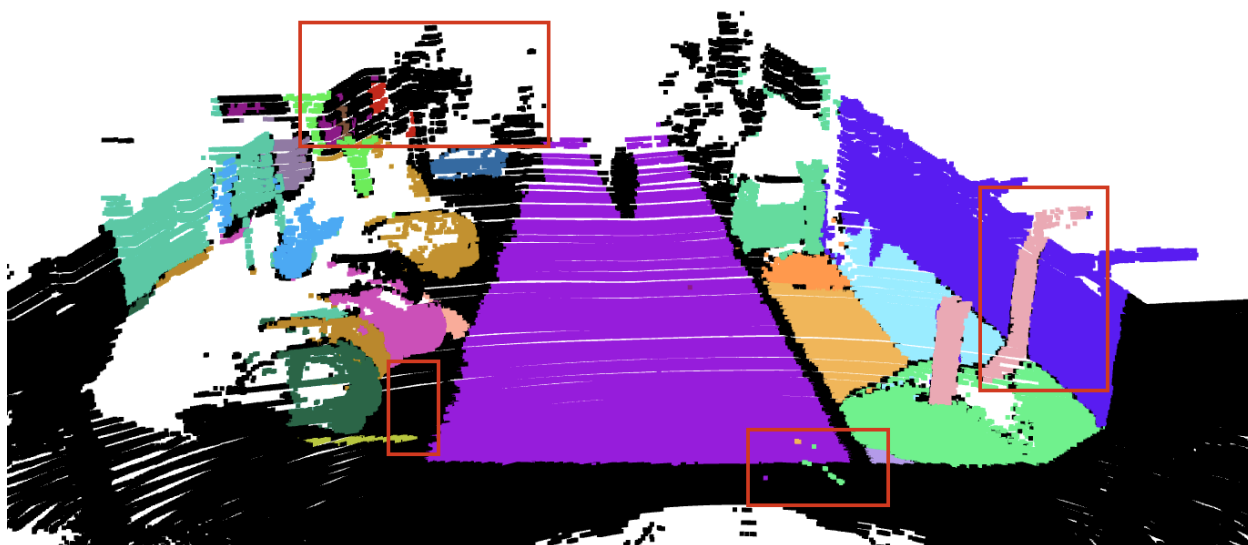


Рис. 4: Ассоциация с пикселями исходной карты

Рассмотренные проблемы делают карту непригодной для кластеризации. Во-первых, облако содержит несколько сотен тысяч точек, что делает невозможным вычисление и хранение матрицы расстояний, необходимой для кластеризации. Во-вторых, крупный фрагмент карты не представляет ценности для сегментации (шум, скрытые и не размеченные на снимках точки), однако будет рассмотрен алгоритмом GraphCut наравне с остальными точками, что может сказаться на точности разметки.

Карта прошла все этапы обработки, описанные в разделе 3.3. Для дальнейшего рассмотрения были оставлены только близкие к датчикам точки, которые удалены от них не более чем на 18 метров. Результат раскраски точек подготовленной карты в цвета соответствующих объектов снимка №20 представлен на рисунке 5. Как можно заметить, в ходе

обработки общее количество точек уменьшилось в несколько десятков раз за счет вокселизации облака. При этом возросла значимость для кластеризации оставшихся точек за счет удаления шума, а также невидимых с данного положения, удаленных от датчиков и не являющихся частью объектов точек.

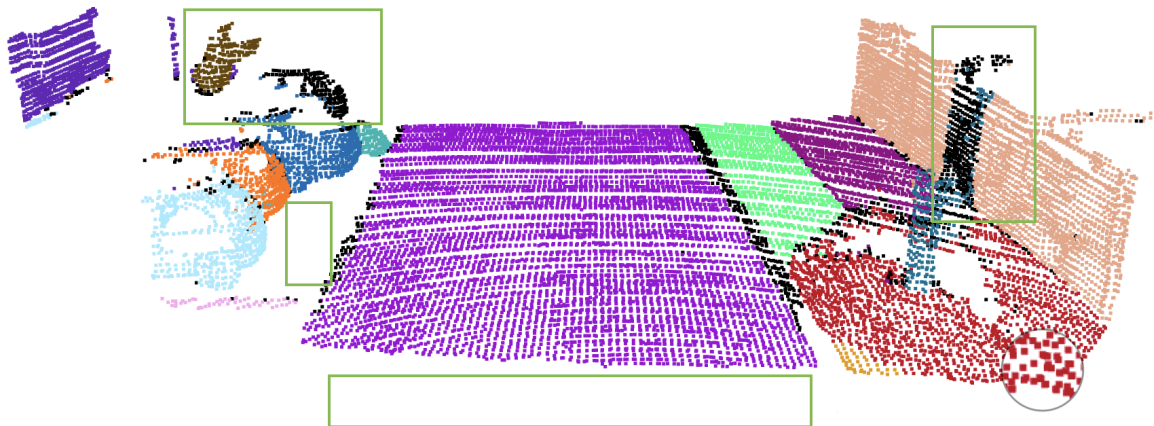


Рис. 5: Ассоциация с пикселями подготовленной к разметке карты

4.3.2. Выбор оптимальных значений гиперпараметров

В разработанном алгоритме есть несколько параметров, значения которых влияют на генерируемую разметку. К ним относятся веса α и β в функции расстояния между точками (пункт 3.4.1) и параметр T , который определяет глубину рекурсии GraphCut (пункт 3.4.3).

В рамках настоящего эксперимента было рассмотрено 392 конфигурации параметров: всевозможные тройки $\alpha \in \{1, 2, 3, 4, 5, 6, 7\}$, $\beta \in \{1, 2, 3, 4, 5, 6, 7\}$, $T \in \{0.001, 0.005, 0.01, 0.02, 0.03, 0.04, 0.05, 0.06\}$. Выбор границ обусловлен следующим. С ростом значений α и β даже физически близкие и похожие по маскам точки в итоге «отдаляются» друг от друга. Левая граница параметра T представляет собой значение по умолчанию в реализации Ncut для обработки изображений в пакете scikit-image¹⁷, правая — значение, рекомендуемое авторами Ncut [21].

¹⁷https://github.com/scikit-image/scikit-image/blob/main/skimage/graph/_graph_cut.py (дата обращения: 24-05-2024)

По результатам сегментации первых 40 облаков датасета KITTI были отобраны конфигурации, которые удовлетворяют хотя бы одному из следующих условий (значения метрик указаны в %):

- $FScore > 75$;
- $recall > 94$ and $FScore > 68$;
- $precision > 75$ and $FScore > 68$.

Пороговые значения взяты на основе анализа средних результатов по запускам. Из 24 подходящих конфигураций некоторые отличались друг от друга значением только одного параметра. В части таких случаев для дальнейшего рассмотрения была оставлена только одна конфигурация, которая демонстрирует лучшие метрики. Для каждой из оставшихся 12 конфигураций была проведена дополнительная разметка 168 облаков. По результатам сегментации были отобраны пять лучших наборов значений параметров, которые были протестированы еще на 1500 облаках KITTI. Результаты приведены в таблице 1.

Значения параметров	precision			recall			FScore		
	avg	100%	0%	avg	100%	0%	avg	100%	0%
$\alpha = 2, \beta = 5, T = 0.05$	67.2	49.4	17.7	61.4	40.3	17.7	61.4	28.9	17.7
$\alpha = 3, \beta = 5, T = 0.04$	65.3	39.7	12.9	69.2	48.3	12.9	64.4	27.7	12.9
$\alpha = 5, \beta = 2, T = 0.03$	59.2	30.0	12.9	68.4	45.7	12.9	60.5	21.1	12.9
$\alpha = 5, \beta = 5, T = 0.02$	62.6	27.1	5.4	79.2	60.3	5.4	67.2	21.7	5.4
$\alpha = 6, \beta = 4, T = 0.02$	58.6	22.6	3.7	81.9	62.6	3.7	65.7	20.3	3.7

Таблица 1: Результаты разметки 1500 облаков KITTI, средние значения метрик в %, доли 100% и 0% значений метрик указаны в % от всех запусков

Таким образом, в связи с наилучшим значением метрики FScore и небольшим процентом неуспешной сегментации (доля 0%), конфигурация параметров $\alpha = 5, \beta = 5, T = 0.02$ была выбрана как оптимальная.

4.3.3. Сравнение разработанного алгоритма с существующими решениями

Многие алгоритмы сегментации лидарных данных ориентированы на разметку в реальном времени, что означает сегментацию *отдельных* лидарных облаков. Разработанный алгоритм сегментирует *плотные* облака, с которыми приходится работать разметчикам. В связи с этим для сравнения рассматриваются только те методы, которые могут работать с подготовленной плотной картой. Среди рассмотренных в пункте 2.1 методов сегментации лидарных облаков удовлетворяет данному требованию и имеет доступную реализацию только один — HDBSCAN [4], который также используется в одних из последних разработанных инструментах сегментации [17, 23]. С этим методом и будет проведено сравнение.

В таблице 2 приведены результаты сегментации подготовленных плотных облаков датасета KITTI двумя алгоритмами: HDBSCAN с его параметрами по умолчанию, который не использует информацию с изображений, и разработанный алгоритм с оптимальными значениями гиперпараметров ($\alpha = \beta = 5$, $T = 0.02$). Визуализация части результатов сегментации представлена на рисунке 6.

Метод сегментации	precision			recall			F1Score		
	avg	100%	0%	avg	100%	0%	avg	100%	0%
HDBSCAN	17.4	12.4	77.4	8.4	1.2	77.4	10.4	1.1	77.4
Ours	62.1	25.4	3.7	82.2	60.1	3.7	67.9	19.8	3.7

Таблица 2: Сравнение сегментации 4540 облаков датасета KITTI разными методами, средние значения метрик в %, доли 100% и 0% значений метрик указаны в % от всех запусков

Исходя из результатов, можно сделать вывод, что использование сегментации изображений в качестве первичной разметки и в дальнейшем одного из признаков кластеризации превосходит сегментацию классическим алгоритмом, который работает только с облаками, по всем метрикам.

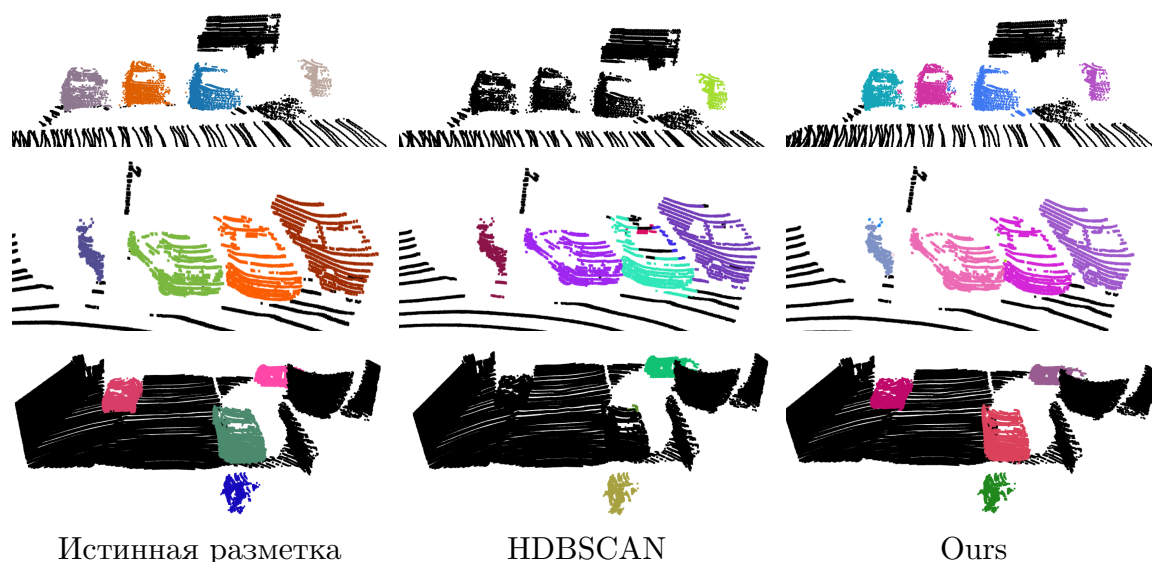


Рис. 6: Результаты сегментации некоторых облаков набора данных KITTI различными алгоритмами

4.4. Вывод

Результаты экспериментального сравнения показывают, что много-ступенчатая обработка облака перед кластеризацией позволяет сократить объем карты в несколько десятков раз с подчеркиванием наиболее значимых для сегментации признаков. Корректно вычисленная матрица расстояний для точек облака с оптимальными значениями весов существенно влияет на результат сегментации — при равной значимости физического расстояния и совпадения первичной разметки признак кластеризации получается наиболее естественным в контексте решаемой задачи, что позитивно влияет на результаты работы GraphCut. Зависимость глубины рекурсии GraphCut от параметра позволяет регулировать детальность генерируемой разметки, при этом в рамках работы удалось подобрать значение параметра, при котором количество и состав кластеров близки к ручной разметке для большинства облаков. Сравнение с существующими решениями, которые в своей работе не используют информацию с изображений, показало превосходство разработанного алгоритма как в точности, так и в полноте разметки.

Заключение

При выполнении данной работы были достигнуты следующие результаты.

1. В рамках обзора рассмотрены существующие алгоритмы сегментации облаков точек, метрики для оценки качества их разметки, выявлен ряд наиболее популярных наборов данных для обучения моделей и тестирования решений.
2. Предложен и реализован алгоритм разметки облаков точек на основе сегментации изображений¹⁸. Реализован набор модульных тестов, создана система непрерывной интеграции на основе GitHub Actions.
3. Определены оптимальные значения гиперпараметров алгоритма и исследовано их влияние на генерируемую сегментацию.
4. Проведено экспериментальное сравнение разработанного алгоритма с ручной разметкой и существующими алгоритмами кластеризации.

¹⁸<https://github.com/prime-slam/lidar-labelling> (дата обращения: 17-04-2024)

Список литературы

- [1] Bogoslavskyi Igor, Stachniss Cyrill. Fast range image-based segmentation of sparse 3D laser scans for online operation // 2016 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS) / IEEE. — 2016. — P. 163–169.
- [2] Boykov Yuri, Funka-Lea Gareth. Graph cuts and efficient ND image segmentation // International journal of computer vision. — 2006. — Vol. 70, no. 2. — P. 109–131.
- [3] Boykov Yuri Y, Jolly M-P. Interactive graph cuts for optimal boundary & region segmentation of objects in ND images // Proceedings eighth IEEE international conference on computer vision. ICCV 2001 / IEEE. — Vol. 1. — 2001. — P. 105–112.
- [4] Campello Ricardo JGB, Moulavi Davoud, Sander Jörg. Density-based clustering based on hierarchical density estimates // Pacific-Asia conference on knowledge discovery and data mining / Springer. — 2013. — P. 160–172.
- [5] Curved-voxel clustering for accurate segmentation of 3D LiDAR point clouds with real-time performance / Seungcheol Park, Shuyu Wang, Hunjung Lim, U Kang // 2019 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS) / IEEE. — 2019. — P. 6459–6464.
- [6] Drive&segment: Unsupervised semantic segmentation of urban scenes via cross-modal distillation / Antonin Vobecky, David Hurych, Oriane Siméoni et al. // European Conference on Computer Vision / Springer. — 2022. — P. 478–495.
- [7] ElC-OIS: Ellipsoidal Clustering for Open-World Instance Segmentation on LiDAR Data / Wenbang Deng, Kaihong Huang, Qinghua Yu et al. // arXiv preprint arXiv:2303.04351. — 2023.

- [8] 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.
- [9] Image-to-lidar self-supervised distillation for autonomous driving data / Corentin Sautier, Gilles Puy, Spyros Gidaris et al. // Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition. — 2022. — P. 9891–9901.
- [10] Learning 3D semantic segmentation with only 2D image supervision / Kyle Genova, Xiaoqi Yin, Abhijit Kundu et al. // 2021 International Conference on 3D Vision (3DV) / IEEE. — 2021. — P. 361–372.
- [11] Li Rong, Cao Anh-Quan, de Charette Raoul. COARSE3D: Class-Prototypes for Contrastive Learning in Weakly-Supervised 3D Point Cloud Segmentation // arXiv preprint arXiv:2210.01784. — 2022.
- [12] Martens Jan, Blut Timothy, Blankenbach Jörg. Cross domain matching for semantic point cloud segmentation based on image segmentation and geometric reasoning // Advanced Engineering Informatics. — 2023. — Vol. 57. — P. 102076.
- [13] Rozenberszki David, Litany Or, Dai Angela. UnScene3D: Unsupervised 3D Instance Segmentation for Indoor Scenes // arXiv preprint arXiv:2303.14541. — 2023.
- [14] SLIC superpixels compared to state-of-the-art superpixel methods / Radhakrishna Achanta, Appu Shaji, Kevin Smith et al. // IEEE transactions on pattern analysis and machine intelligence. — 2012. — Vol. 34, no. 11. — P. 2274–2282.
- [15] Scalability in perception for autonomous driving: Waymo open dataset / Pei Sun, Henrik Kretzschmar, Xerxes Dotiwalla et al. // Proceedings of the IEEE/CVF conference on computer vision and pattern recognition. — 2020. — P. 2446–2454.

- [16] SegContrast: 3D point cloud feature representation learning through self-supervised segment discrimination / Lucas Nunes, Rodrigo Marcuzzi, Xieyuanli Chen et al. // IEEE Robotics and Automation Letters. — 2022. — Vol. 7, no. 2. — P. 2116–2123.
- [17] Segment Any Point Cloud Sequences by Distilling Vision Foundation Models / Youquan Liu, Lingdong Kong, Jun Cen et al. // arXiv preprint arXiv:2306.09347. — 2023.
- [18] Segment anything / Alexander Kirillov, Eric Mintun, Nikhila Ravi et al. // arXiv preprint arXiv:2304.02643. — 2023.
- [19] Segment everything everywhere all at once / Xueyan Zou, Jianwei Yang, Hao Zhang et al. // arXiv preprint arXiv:2304.06718. — 2023.
- [20] Semantickitti: A dataset for semantic scene understanding of lidar sequences / Jens Behley, Martin Garbade, Andres Milioto et al. // Proceedings of the IEEE/CVF international conference on computer vision. — 2019. — P. 9297–9307.
- [21] Shi Jianbo, Malik Jitendra. Normalized cuts and image segmentation // IEEE Transactions on pattern analysis and machine intelligence. — 2000. — Vol. 22, no. 8. — P. 888–905.
- [22] Unal Ozan, Dai Dengxin, Van Gool Luc. Scribble-supervised lidar semantic segmentation // Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition. — 2022. — P. 2697–2707.
- [23] Unsupervised class-agnostic instance segmentation of 3D LiDAR data for autonomous vehicles / Lucas Nunes, Xieyuanli Chen, Rodrigo Marcuzzi et al. // IEEE Robotics and Automation Letters. — 2022. — Vol. 7, no. 4. — P. 8713–8720.
- [24] Yuan Xia, Mao Yangyukun, Zhao Chunxia. Unsupervised segmentation of urban 3d point cloud based on lidar-image // 2019 IEEE International Conference on Robotics and Biomimetics (ROBIO) / IEEE. — 2019. — P. 2565–2570.

- [25] Zhou Qian-Yi, Park Jaesik, Koltun Vladlen. Open3D: A Modern Library for 3D Data Processing // arXiv:1801.09847. — 2018.
- [26] a Semantic 3d Point Cloud Segmentation Approach Based on Optimal View Selection for 2d Image Feature Extraction / A Adam, L Grammatikopoulos, G Karras et al. // The International Archives of the Photogrammetry, Remote Sensing and Spatial Information Sciences. — 2019. — Vol. 42. — P. 9–14.
- [27] An image-based deep learning workflow for 3D heritage point cloud semantic segmentation / Eugénio Pellis, Arnadi Murthy, Andrea Masiero et al. // 9th International Workshop 3D-ARCH” 3D Virtual Reconstruction and Visualization of Complex Architectures”. 2–4 March 2022, Mantua, Italy / ISPRS. — Vol. 46. — 2022. — P. 426–434.
- [28] nusenes: A multimodal dataset for autonomous driving / Holger Caesar, Varun Bankiti, Alex H Lang et al. // Proceedings of the IEEE/CVF conference on computer vision and pattern recognition. — 2020. — P. 11621–11631.
- [29] A survey of convolutional neural networks: analysis, applications, and prospects / Zewen Li, Fan Liu, Wenjie Yang et al. // IEEE transactions on neural networks and learning systems. — 2021.
- [30] A survey on deep domain adaptation for lidar perception / Larissa T Triess, Mariella Dreissig, Christoph B Rist, J Marius Zöllner // 2021 IEEE intelligent vehicles symposium workshops (IV workshops) / IEEE. — 2021. — P. 350–357.