Федеральное государственное учреждение «Федеральный исследовательский центр «Информатика и управление» Российской академии наук»

На правах рукописи

Муравьев Кирилл Федорович

Исследование методов и разработка алгоритмов топологического картирования и локализации

Специальность 1.2.2 —
«Математическое моделирование, численные методы и комплексы программ»

Диссертация на соискание учёной степени
кандидата технических наук

Научный руководитель:
канд. физ.-мат. наук
Яковлев Константин Сергеевич

Москва — 2025

Оглавление

Стр.

Стр.

Введение

Актуальность темы. Задача одновременного картирования и локализации (ОКЛ) является одной из важнейших для обеспечения навигации робототехнических систем. Ее решение позволяет робототехнической системе определять свое положение в пространстве, не опираясь на системы глобального позиционирования, а также строить карту местности, учитывая изменения среды. Таким образом, успешное решение задачи ОКЛ дает возможность применять робототехнические системы в таких областях, как автоматизированная доставка грузов в условиях нестабильной работы систем спутникового позиционирования, поисково-спасательные операции, мониторинг и патрулирование различных объектов и др.

В настоящее время, как правило, задача ОКЛ решается методами, которые строят карту в виде плотных метрических структур, таких как двумерная сетка занятости или трехмерные воксельные сетки. Однако, в случае, к примеру, автоматизированной доставки, робототехнические системы преодолевают большие расстояния и картируют большие площади. В такой ситуации поддержание плотной метрической карты и коррекция ошибки одометрии требуют значительных затрат памяти и вычислительных ресурсов, что может привести к переполнению памяти, задержке обновления карты, накоплению ошибки позиционирования. Все это может привести к некорректной работе алгоритмов и, как следствие, — к остановке робота либо к столкновению его с препятствиями.

Альтернативным подходом к решению задачи ОКЛ является топологическое картирование и локализация. Идея такого подхода заключается в представлении окружающей среды в виде разреженных топологических структур, таких как граф локаций, вместо плотных метрических структур. За счет разреженности графа такой подход обеспечивает быстрое планирование пути и позволяет избавиться от накопления ошибки позиционирования при долговременной навигации. Таким образом, построение топологической карты и локализация в ней позволят обеспечить эффективную долговременную автономную навигацию робототехнических систем в средах большой площади.

В связи с этим актуальной является проблема разработки алгоритмов построения топологической карты по данным бортовых сенсоров робототех-

нической системы, а также проблема разработки алгоритмов локализации робототехнической системы в топологической карте.

Степень разработанности темы. Существует множество методов, которые используют графовые структуры для создания глобальной геометрической модели окружающей среды (метрической карты). Среди наиболее известных методов такого класса можно выделить ORB-SLAM3, Cartographer и RTAB-Map, которые используют графы позиций для глобальной оптимизации оценки траектории робота и коррекции построенной метрической карты, и методы Voxgraph и GLIM, которые строят граф локальных метрических карт, объединяемых в общую глобальную метрическую карту. Существует довольно обширный класс методов, которые строят совместно метрическую и топологическую карту. Среди таких методов – Hydra, S-graphs+, IncrementalTopo. Общими недостатками подобных методов являются высокие затраты памяти и вычислительных ресурсов при построении плотной глобальной метрической карты пространств большой площади, а также неизбежное накопление ошибки глобального позиционирования при долговременной работе метода.

Развитие технологий глубокого обучения и рост мощности вычислителей привели к появлению обучаемых методов, решающих задачу ОКЛ с помощью построения графа локаций без использования метрических координат. Локализация в таких графах и перемещение между локациями осуществляется нейросетевыми методами. В качестве примеров таких методов можно привести NTS, ETP-Nav, VGM, TSGM. Однако такие методы, как правило, разрабатываются для решения определенной краткосрочной задачи, такой как навигация до одного целевого объекта, и работают преимущественно в симуляционных средах в небольших помещениях. Использование полностью нейросетевых методов при долгосрочной навигации может привести к ложным срабатываниям нейросетевой локализации, и, как следствие, к соединению ребрами удаленных друг от друга локаций и к ошибкам навигации. В работе исследователей из Монреаля представлен обучаемый алгоритм топологического картирования LTVN, пригодный для долгосрочной навигации, однако для начала его работы необходима предварительно построенная карта.

Еще одним направлением развития топологического картирования и локализации являются методы, которые строят глобальную топологическую карту в виде графа локаций с использованием локальной метрической информации. К настоящему моменту разработаны такие методы, предназначенные как для

исследования и картирования помещений, так и для картирования открытых пространств, локализации и навигации по построенной карте в разное время суток.

Таким образом, на настоящий момент разработано множество методов решения задачи картирования и локализации с применением топологических структур, однако отсутствуют методы, обеспечивающие долговременную навигацию как в помещениях, так и на открытых пространствах с низкими затратами вычислительных ресурсов, что обуславливает необходимость данного исследования.

Целью работы является исследование и разработка вычислительно эффективных алгоритмов ОКЛ на основе топологических структур для повышения автономности мобильных робототехнических систем.

Для достижения поставленной цели необходимо было решить следующие **задачи**:

1. Провести анализ существующих методов топологического картирования и локализации.
2. Построить математическую модель задачи топологического картирования и локализации и оценки качества ее решения.
3. Разработать вычислительно эффективный алгоритм топологического картирования и локализации, обладающий высоким качеством локализации и построенной карты.
4. Создать программный комплекс топологического картирования и локализации, провести экспериментальные исследования разработанных алгоритмов с использованием предложенной математической модели в симуляционных средах и на реальных робототехнических системах.

Научная новизна работы состоит в следующем:

Предложен новый алгоритм построения и поддержания топологической карты в реальном времени. Карта представляется в виде графа локаций, не содержащих глобальных метрических координат. Такое представление позволяет выполнять долговременную навигацию без накопления ошибки позиционирования и снижает потребление вычислительных ресурсов и памяти при долговременной навигации.

Предложен двухэтапный алгоритм локализации в построенной карте, основанный на нейросетевых методах распознавания места (локации) и поиске относительной позиции путем сопоставления двумерных сканов местности. В

отличие от аналогичных методов, предложенный подход позволяет фильтровать ложно распознанные нейросетевыми моделями локации и корректировать ошибку одометрии в реальном времени.

Реализован двухуровневый алгоритм планирования пути по построенной карте, который позволяет значительно ускорить планирование пути по сравнению с алгоритмами, использующими глобальную метрическую карту. Проведено экспериментальное исследование комплекса предложенных алгоритмов картирования, локализации и планирования пути в симуляционной среде и на данных с реальных робототехнических систем.

Предложена новая математическая модель оценки качества графов локаций. В отличие от аналогов, опирающихся на глобальные метрические координаты, предложенная модель позволяет оценивать качество графа с точки зрения путевой эффективности в случае отсутствия глобальных метрических координат в локациях графа. Предложена новая модель оценки качества локализации, которая помимо точности вычисления относительной позиции учитывает еще успешность локализации и долю ложно сопоставленных локаций. Проведено экспериментальное исследование известных алгоритмов топологического картирования и локализации с помощью предложенных моделей оценки качества. Показано преимущество разработанного комплекса алгоритмов перед другими современными алгоритмами.

Теоретическая значимость работы обуславливается комплексом разработанных алгоритмов и моделей, которые создают основу как для создания новых топологических методов решения задачи ОКЛ, так и для улучшения существующих.

Практическая значимость работы заключается в реализации разработанных алгоритмов и моделей в виде комплекса программных средств для реальных робототехнических систем. Реализованные алгоритмы могут быть использованы для повышения автономности робототехнических систем различного типа и назначения.

Соответствие диссертации паспорту научной специальности. В соответствии с формулой специальности 1.2.2 «Математическое моделирование, численные методы и комплексы программ» (технические науки) в работе предложена математическая модель задачи топологического картирования и локализации и выполнены разработка, исследование и реализация алгоритмов и методов решения этой задачи. Работа соответствует следующим пунктам

паспорта специальности: п. 2. «Разработка, обоснование и тестирование эффективных вычислительных методов с применением современных компьютерных технологий», п. 3. «Реализация эффективных численных методов и алгоритмов в виде комплексов проблемно-ориентированных программ для проведения вычислительного эксперимента», п. 8. «Комплексные исследования научных и технических проблем с применением современной технологии математического моделирования и вычислительного эксперимента», п. 9. «Постановка и проведение численных экспериментов, статистический анализ их результатов, в том числе с применением современных компьютерных технологий (технические науки)».

Методология и методы исследования. Методы исследования и разработки алгоритмов топологического картирования и локализации основаны на теории графов, теории вероятностей, математической статистике, линейной алгебре, аналитической геометрии, компьютерном зрении, методах разработки и тестирования программного обеспечения для ЭВМ.

Основные положения, выносимые на защиту:

1. Предложена математическая модель задачи топологического картирования и локализации, как построения графа локаций по входным данным с датчиков робота и определения текущей локации в построенном графе и положения внутри локации. Разработана модель оценки качества графа локаций, не требующая наличия глобальных метрических координат в локациях. Предложена новая модель оценки качества локализации в графе локаций.
2. Предложен алгоритм построения и поддержания карты местности в виде графа в реальном времени, обеспечивающий долговременную навигацию. Предложенный алгоритм основан на построении графа локаций без глобальных метрических координат, что позволяет избавиться от накопления ошибки позиционирования и повысить вычислительную эффективность методов автономной навигации.
3. Предложен двухэтапный алгоритм локализации в построенном графе локаций, основанный на глобальном поиске похожих локаций в графе с помощью нейросетевых методов и одновременной фильтрации результатов и нахождении относительной позиции с помощью сопоставления сканов. Предложенный алгоритм позволяет повысить точность

локализации и уменьшить количество ложных сопоставлений текущего положения робота с локациями в графе.

4. Предложенные алгоритмы картирования и локализации реализованы в виде комплекса программных средств и выложены в открытый доступ. Программный комплекс позволяет сохранять и загружать построенные карты, настраивать параметры алгоритмов и запускать реализации алгоритмов на различных робототехнических системах без доработки исходного кода.

Достоверность полученных результатов подтверждается данными численных экспериментов, проведенных в симуляционных средах с помощью разработанных алгоритмов, математических моделей и комплекса программ, а также успешной апробацией разработанных алгоритмов и комплекса программ на данных с реальных робототехнических систем.

Апробация работы. Основные результаты работы докладывались на следующих научных конференциях: The 9th International Conference on Interactive Collaborative Robotics (ICR 2024), The 16th International Conference on Machine Vision (ICMV 2023), The 7th International Conference on Interactive Collaborative Robotics (ICR 2022), XX Всероссийская научно-практическая конференция «Перспективные системы и задачи управления» (Домбайская конференция 2025), XIV Всероссийское совещание по проблемам управления (ВСПУ 2024). Результаты были получены в процессе выполнения работ по грантам: №075-15-2020-799 «Методы построения и моделирования сложных систем на основе интеллектуальных и суперкомпьютерных технологий, направленные на преодоление больших вызовов» и №075-15-2024-544 «Математические модели и численные методы как основа для разработки робототехнических комплексов, новых материалов и интеллектуальных технологий конструирования» Министерства науки и высшего образования РФ.

Личный вклад. Все положения, выносимые на защиту и изложенные в диссертации, принадлежат лично автору. Постановка задач и обсуждение результатов проводились совместно с научным руководителем. В [1] автором проведено подробное экспериментальное исследование двух современных алгоритмов ОКЛ и проведен анализ их эффективности. В [2] автором предложен набор критериев для оценки качества графа локаций и проведено обширное экспериментальное исследование современных топологических алгоритмов ОКЛ с использованием предложенного набора критериев. В [3] автором предложен

алгоритм построения и обновления графа локаций и проведены численные эксперименты. В [4; 5] автором доработан предложенный ранее алгоритм построения и обновления графа локаций, а также предложен алгоритм локализации робота в графе локаций и проведено экспериментальное исследование комплекса разработанных алгоритмов. В [6] автором предложен подход к планированию пути в графе локаций и проведено экспериментальное исследование предложенного подхода.

Публикации. Основные результаты по теме диссертации изложены в 10 печатных изданиях, в том числе 3 работы опубликованы в изданиях из списка ВАК категории К1 и приравненных к ним, из которых 2 – индексируются в Scopus (Q1), 6 работ опубликованы в трудах конференций, из которых 4 – индексируются в Scopus. Зарегистрирована 1 программа для ЭВМ.

Объем и структура работы. Диссертация состоит из введения, 5 глав, заключения и 1 приложения. Полный объём диссертации составляет 125 страниц, включая 40 рисунков и 19 таблиц. Список литературы содержит 101 наименование.

Глава 1. Обзор и анализ методов и алгоритмов топологического картирования и локализации

Автономная навигация является одним из ключевых аспектов для функционирования мобильных роботов и беспилотных транспортных средств. Задача автономной навигации обычно формулируется как достижение роботом заданной целевой точки. Для успешного достижения целевой точки необходимо решить следующие подзадачи навигации:

1. Определение положения робота и целевой точки;
2. Построение маршрута от положения робота до целевой точки;
3. Движение робота вдоль построенного маршрута с избеганием столкновений с препятствиями.

Внешнее позиционирование робота (например, с помощью глобальных навигационных спутниковых систем, далее – ГНСС) зачастую бывает недоступно или нестабильно (в частности, при навигации внутри помещений или в плотной городской застройке). Карта местности, на которой выполняется навигация, может отсутствовать или быть устаревшей. Таким образом, возникают задачи локализации (определения положения робота на карте) и картирования (создания карты местности, используемой для локализации и построения маршрутов) по данным с бортовых датчиков робота. В данной главе рассматриваются методы и алгоритмы решения указанных задач с использованием данных с различных бортовых датчиков. Оцениваются преимущества и недостатки методов, их применимость в различных условиях навигации.

1.1 Восприятие окружающей среды роботами

В процессе автономной навигации роботам приходится решать комплекс задач, таких как картирование окружающего пространства, определение своего местоположения на карте, планирование пути до целевой точки и движение вдоль спланированного пути. Для построения карты и оценки местоположения робота в ней могут использоваться данные с различных типов датчиков. Чаще всего используются следующие виды датчиков:

- Монокулярные камеры, генерирующие трехканальное изображение путем проецирования предметов окружающего пространства на плоскость матрицы. Они обладают неограниченной дальностью действия, однако зависят от освещенности окружающей среды и по данным с ним невозможно определить расстояния до окружающих предметов.
- Стереокамеры – пара камер, расположенных на определенном расстоянии друг от друга. Стереокамеры так же чувствительны к освещенности, однако позволяют оценивать расстояние до окружающих предметов путем вычисления диспаритета (разности в положении проекций объектов) между изображениями с левой и с правой камер. Таким образом, на практике с помощью стереокамер можно оценивать расстояния до нескольких метров.
- Камеры глубины, или RGB-D (от англ. Red, Green, Blue, Depth), помимо цветного трехканального изображения дают его карту глубин – матрицу проекций расстояний до изображенных объектов на главную оптическую ось. Карта глубин строится с помощью инфракрасной сетки, излучаемой камерой, что позволяет оценивать расстояния даже на изображениях без четких контуров объектов в пределах нескольких метров.
- Лазерные сканеры, или лидары (от англ. Light Detection and Ranging) – позволяют оценить расстояния до объектов вокруг робота. Так как принцип работы основан на измерении времени отражения испускаемого датчиком лазерного луча, то лидары не зависят от внешнего освещения, однако дальность их действия составляет не более 100-200 м.
- Датчики инерциальной навигационной системы (гироскопы, акселерометры, магнитометры) позволяют измерить ускорение и поворот робота и таким образом оценить его перемещение в пространстве. Однако такой способ оценки перемещения робота имеет значительную погрешность.
- Датчики вращения колес (энкодеры) позволяют отследить число оборотов и угол поворота каждого колеса и таким образом оценить перемещение робота в пространстве. Такой способ позволяет довольно точно вычислить пройденное расстояние при движении по прямой без проскальзывания, однако обладает большой погрешностью в случае поворотов робота.

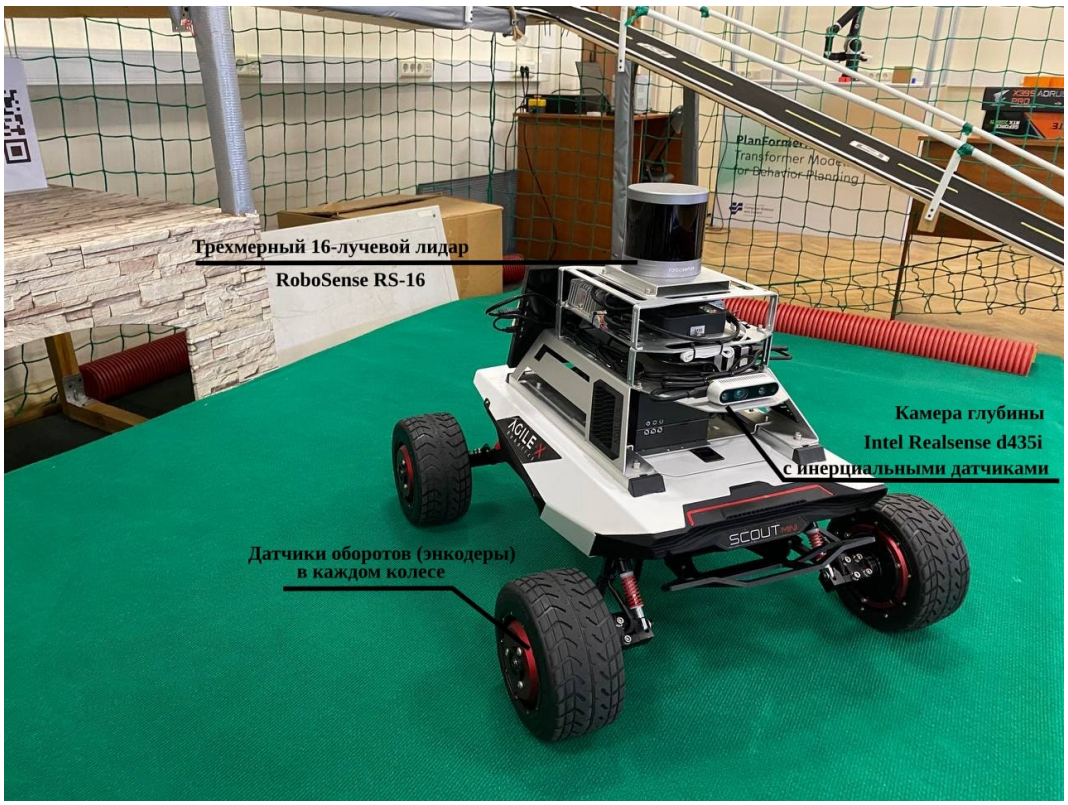

Рисунок 1.1 — Пример робототехнической системы AgileX Scout Mini с лидаром, камерой глубины и энкодерами колес.

Пример робототехнической системы с набором датчиков для картирования и определения положения в пространстве показан на рисунке 1.1. Набор датчиков включает в себя трехмерный лазерный сканер, камеру глубины, совмещенную с инерциальной навигационной системой, а также датчики вращения колес.

Данные об окружающем мире с монокулярных и стереокамер подаются на вход алгоритмам в виде цветных трехканальных изображений. Данные с камер глубины подаются на вход в виде карт глубин, содержащих расстояния в метрах. Изображения и карты глубин требуют дополнительных методов для интеграции данных с них в карту – например, преобразование карты глубин в облако точек с помощью обратной проекции и выделение особых точек на изображении. Трехмерные многолучевые лидары выдают набор расстояний для каждого луча, которые легко преобразуются в готовое облако точек, картирующее участок окружающей среды вокруг робота. Такие облака точек, снятые в различных локациях, могут быть объединены в одну общую карту (глобальное облако точек), а также спроецированы на плоскость для создания карты препятствий. Пример данных с датчиков робота (изображение, карта глубины, облако точек) изображен на рисунке 1.2.

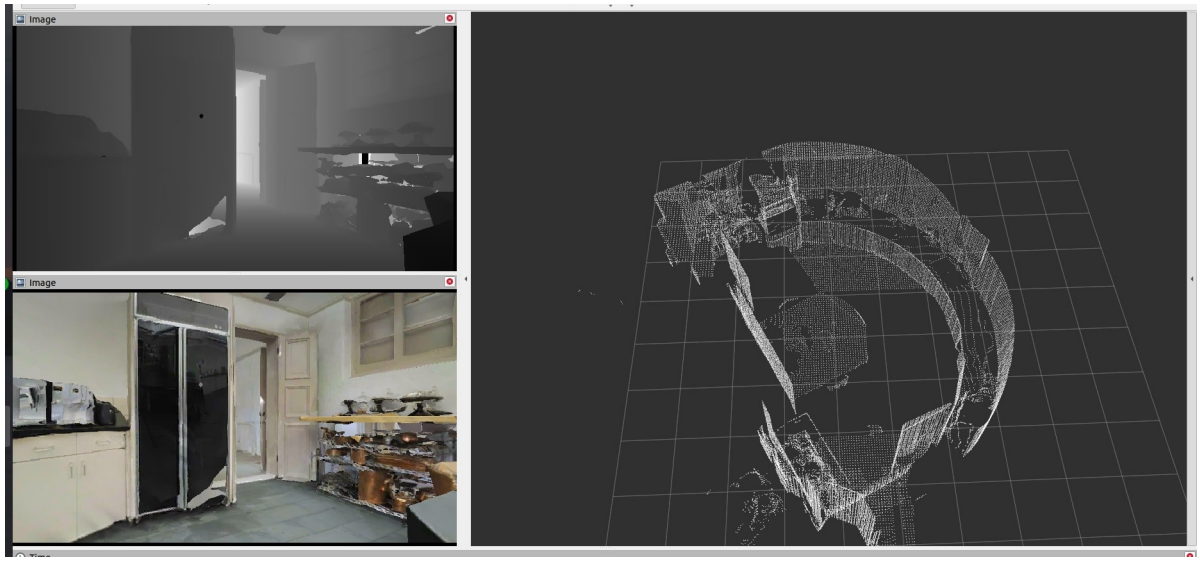

Рисунок 1.2 — Пример данных с датчиков робота: изображение (слева внизу), карта глубин (слева вверху), облако точек (справа).

Для определения положения робота в пространстве, как правило, используются данные одометрии – оценки перемещения и/или скорости робота. Одометрия может вычисляться по датчикам инерциальной навигационной системы (далее – ИНС) путем интегрирования показаний акселерометра и гироскопа и соотнесения их с показаниями магнетометра. Также одометрия может вычисляться по датчикам вращения колес (в случае робота с дифференциальным приводом угол поворота определяется по разности вращения левых и правых колес), либо по данным с камер (визуальная одометрия), либо по данным с лидаров (лидарная одометрия).

Методы вычисления визуальной одометрии (например, ORB-SLAM3 [10]), как правило, основаны на извлечении особых точек из изображений и отслеживании перемещения робота по перемещению особых точек от кадра к кадру. Методы вычисления лидарной одометрии (например, LOAM [11]) могут быть основаны на вычислении особых точек в облаках точек, либо на вычислении прямых и плоскостей (если движение происходит внутри помещений), либо на полном сопоставлении сканов или других методах. Распространение получили и алгоритмы, комплексирующие различные источники для вычисления одометрии. Например, алгоритм VINS-Fusion [12] использует данные инерциальной навигационной системы для уточнения визуальной одометрии, а алгоритм GLIM [13] может использовать инерциальные данные для уточнения лидарной одометрии. Однако одометрия в любом случае имеет некоторую относительную ошибку определения смещения и поворота. На больших расстояниях и/или при

большом количестве поворотов накопление ошибки приводит к невозможности точной оценки положения робота по данным одометрии. Для коррекции накапливающейся ошибки, как правило, используется привязка робота к карте (локализация в карте).

Карты, используемые на роботах, представляют собой модели окружающей среды и используются для локализации робота и планирования пути до целевой точки. Модель окружающей среды может представляться в виде различных структур, среди которых преобладают геометрические структуры. В частности, нередко используется представление среды в виде двумерной или трехмерной сетки занятости. В такой сетке каждая ячейка имеет значение 0 или 1 – значения в ячейках определяют, свободен или занят для проезда соответствующий квадрат поверхности или куб пространства. При навигации по неровной поверхности могут использоваться карты высот – матрицы, в которых каждая ячейка содержит высоту соответствующего квадрата поверхности. Такие карты высот можно преобразовать в сетки занятости для конкретного робота с учетом его характеристик (габариты, радиус колес, дорожный просвет и т.д.).

Еще одним распространенным способом представления местности является облако точек. При наличии информации о позиции робота облака точек с лидара легко объединяются в общее облако точек, которое и представляет собой трехмерную модель окружающей среды со всеми объектами. Плотное облако точек для среды большой площади занимает значительные объемы памяти, и для экономии памяти и ускорения алгоритмов локализации может быть использовано разреженное или дискретизованное облако точек. Для наиболее детального представления окружающей среды могут использоваться трехмерные полигональные сетки (англ. 3D Mesh), однако такой способ представления является наиболее затратным по памяти. Виды метрических представлений окружающей среды представлены на рисунке 1.3.

Метрические карты удобны для алгоритмической обработки и позиционирования робота в пространстве $\mathbb{R}^2$ или $\mathbb{R}^3$. Однако при большой площади среды такие карты требуют значительных объемов памяти для хранения и значительных вычислительных ресурсов для построения и обработки. Альтернативным способом представления окружающей среды является топологический подход, при котором моделью среды является граф. Вершинами графа могут быть локации (области пространства), каждая из которых представлена описанной выше геометрической структурой или вектором признаков. Также вершинами могут

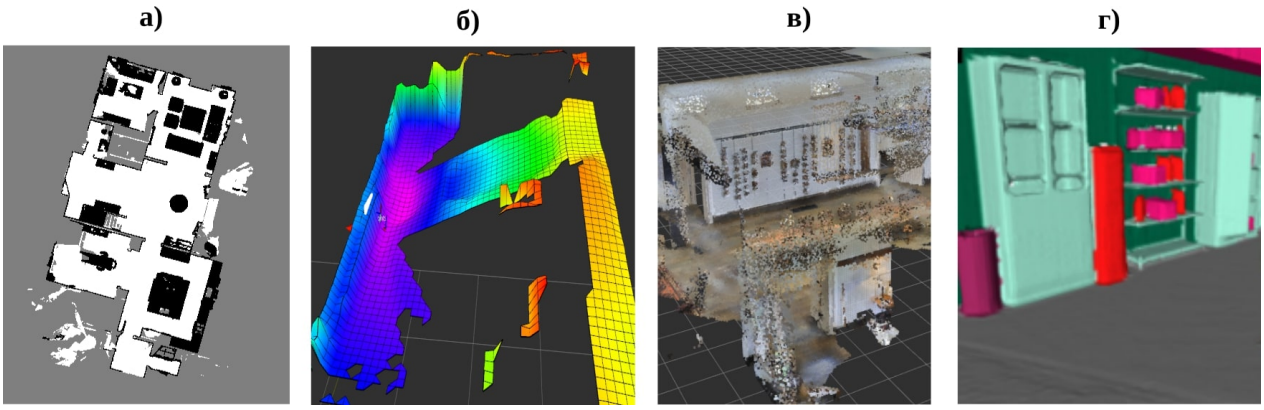

Рисунок 1.3 — Виды метрических представлений среды, используемых на роботах: (а) двумерная сетка занятости; (б) карта высот; (в) трехмерное облако точек; (г) трехмерная полигональная сетка.

быть позиции выделенных ключевых кадров, в таком случае каждая из вершин будет представлена наблюдением с робота. Методы и алгоритмы построения метрических и топологических карт и локализации в них подробно рассмотрены ниже.

1.2 Методы и алгоритмы картирования

Методы и алгоритмы картирования подразделяются на два больших класса в соответствии с картами, которые они строят: метрические и топологические. Топологические методы, в свою очередь, подразделяются на две большие группы: онлайн-методы и офлайн-методы. Первая группа методов строит топологическую карту с нуля в реальном времени по наблюдениям с бортовых датчиков робота. Вторая группа строит топологическую карту по заранее собранному набору наблюдений или по предварительно построенной глобальной метрической карте. Среди онлайн-методов топологического картирования встречается множество методов, использующих глобальные метрические координаты (т.н. топометрические методы) и чисто топологические методы, не опирающиеся на глобальные метрические координаты. Чисто топологические методы могут опираться на локальные метрические координаты (например, положение робота относительно центра локации) или не использовать координаты вообще. Схема классификации методов картирования представлена на рисунке 1.4. Подробный обзор классов методов приведен ниже.

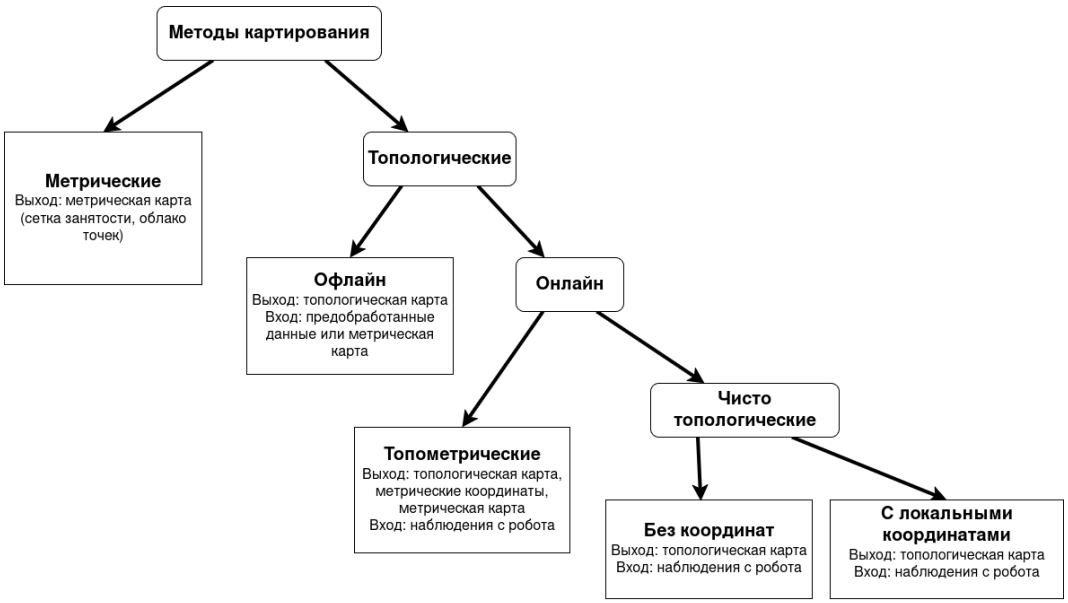

Рисунок 1.4 — Классификация методов картирования.

Метрические методы История методов автоматического картирования берет свое начало в 1980-х годах [14]. Первые методы картирования были метрическими. Например, в работе [15] описан метод построения карты занятости (Occupancy Grid) в виде двумерной сетки по данным с сонаров — акустических сенсоров, определяющих расстояния до объектов посредством эхолокации. Метод [15] позволил получить плотные карты окружающей среды, в которых области классифицировались как свободные, занятые и неизвестные. Такие карты были пригодны для автономной навигации и планирования на высоком уровне. Для построения карт использовалась интерпретация измерений дальности с сонарных датчиков, моделирование информации о занятости с помощью вероятностных профилей и последующее проецирование информации на двумерную карту. Пример такой двумерной карты занятости показан на рисунке 1.5.

Методы картирования, основанные на сетках занятости, впоследствии были использованы во множестве робототехнических систем. В современных методах картирования (например, RTAB-Map [16]) используется построение сетки занятости с вероятностными оптимизациями. Двумерные сетки занятости являются интуитивно понятным представлением окружающей среды и требуют небольшого количества памяти для хранения. По ним можно легко планировать маршруты с помощью алгоритмов поиска пути в графе, например, алгоритма A* [17]. Однако при больших размерах сетки (1000 ячеек и более) планирование маршрута занимает значительное время, что может привести к затруднени-

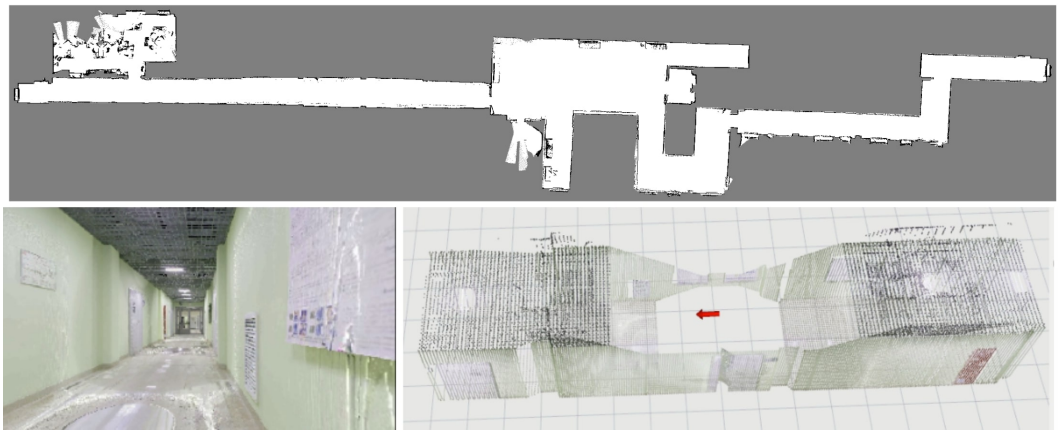

Рисунок 1.5 — Пример карты занятости помещения, состоящего из комнат, холлов и коридоров (сверху, белым показаны свободные области, черным – занятые, серым – неизвестные). Внизу слева показано изображение с камеры робота, внизу справа – облако точек с лидара.

ям при навигации роботов. Таким образом, при навигации в средах большого размера необходимо либо увеличивать размеры ячейки сетки занятости, либо использовать другие методы.

Двумерные сетки занятости являются удобным и легким инструментом представления окружающей среды. Однако в случае навигации малого беспилотного летательного аппарата или наземного робота по неровной местности, 2D-карты не являются достаточно надежными из-за неполной информации о поверхности и отсутствия отображения уровней. Для полного отображения окружающего пространства может быть использована трехмерная сетка занятости, состоящая из вокселей (небольших кубиков). Значения в кубиках показывают, свободна для прохода или занята соответствующая область пространства. Первые методы трехмерного картирования появились в конце 1980-х годов [18]. В методе [18] каждая ячейка воксельной сетки помечается как свободная, занятая или неизвестная. Обновление сетки по данным с сенсоров происходит с помощью вероятностных алгоритмов и методов сглаживания.

С развитием лидарных технологий в 1990-х годах появились методы построения трехмерной метрической карты в виде облака точек (Point Cloud) — набора точек с координатами (x, y, z) и вспомогательными атрибутами (цвет и т. д.). Один из первых таких методов был представлен в работе [19] в 2000 году. Методы построения карт в виде облака точек популярны по сей день, поскольку облака точек поступают напрямую с сенсоров робота (лидаров или камер глубины). Глобальное облако точек строится в методе RTAB-MAP и дру-

гих современных методах метрического картирования, например, GLIM [13], ORB-SLAM3 [10], Cartographer [20]. Еще одним форматом данных, часто используемым в современных методах 3D-картирования, является поле евклидовых расстояний со знаком (Euclidean Signed Distance Field, ESDF). ESDF представляет собой разновидность воксельной сетки, в кубиках которой записаны расстояния от центра кубика до ближайшего препятствия. Такой формат карты позволяет строить безопасные пути для роботов. Карта в виде ESDF строится в том числе в методе Voxblox [21], который является составной частью других методов картирования.

Топологические методы Метрические карты удобны для алгоритмической обработки и понятны для человека, тем не менее, они обладают рядом существенных недостатков. Во-первых, метрические карты пространств большого размера занимают большие объемы памяти (особенно в виде трехмерных воксельных сеток), а их обработка требует значительных вычислительных затрат. Во-вторых, планирование пути по сеткам большой размерности требует значительных вычислительных ресурсов и занимает достаточно много времени. В-третьих, из-за несовершенства датчиков и накопления ошибки, метрические карты большого размера, как правило, получаются неточными. И в-четвертых, метрическое представление местности хоть и понятно людям, но не используется людьми при навигации на нативном уровне. Когда человек идет по улице или по коридору здания, он не считает количество метров, которое ему нужно пройти, и не задает направление по компасу. Он использует визуальные ориентиры и представление о расположении улиц или коридоров относительно друг друга, т. е. топологическое представление местности.

Автоматизированное представление топологических карт может позволить устранить вышеописанные недостатки при навигации роботов. Работы по топологическому картированию ведутся с начала 1990-х годов. Среди первых методов топологического картирования можно выделить работы [22—24]. В работе [22] строится топологическая карта в процессе исследования неизвестной местности, и затем по топологической карте строится метрическая. Процесс построения карты включает в себя детекцию похожих мест и замыкание циклов. В работе [23] представлен и опробован на реальном роботе полный алгоритм навигации с построением топологической карты. В работе [24] для повышения точности построения топологической карты использовались технологии

машинного обучения. В настоящее время разработано множество методов и алгоритмов построения топологической карты, работающих в реальном времени или на основе предварительно собранного и обработанного набора данных. Обзор основных современных методов и алгоритмов приведен ниже.

Офлайн-методы топологического картирования Одним из наиболее известных офлайн-методов построения топологической карты является метод TopoMap [25], разработанный в 2018 году. В качестве предварительно построенной карты на вход методу подается глобальное разреженное облако точек, которое может быть получено, например, с помощью алгоритма ORB-SLAM [10]. По этому облаку точек строится трехмерная воксельная сетка с помощью трассировки лучей от позиции робота до координаты точки и метода Voxblox [21]. Полученное с помощью метода Voxblox усеченное поле расстояний со знаком (англ. Truncated Signed Distance Field, TSDF) дополнительно фильтруется по порогу, равному 0.9, также удаляются небольшие изолированные группы вокселей.

По полученной воксельной сетке строится метрическая карта в виде глобальной сетки занятости и топологическая карта в виде графа выпуклых кластеров свободного пространства. Пример исходного разреженного облака точек и построенной по нему топологической карты показан на рисунке 1.6. Выпуклые кластеры строятся так: за центр принимается случайная точка на траектории робота, затем вокруг нее “раздувается” кластер с помощью метода главных компонент и итеративного расширения вдоль его главной оси. Два кластера объединяются в один, если их совместная выпуклая оболочка содержит не более чем 1-5% вокселей препятствия. Эксперименты, проведенные на симуляционных сценах помещений, показали, что построенная топологическая карта обеспечивает в 2000 раз более быстрое планирование пути, чем метрическая карта.

В работе [26] опубликован еще один офлайн-метод, позволяющий построить топологическую карту окружающей среды масштабов города. Для построения топологической карты используются данные общедоступных глобальных карт города (например, OpenStreetMap [1]). Вершинами топологической карты являются объекты городской инфраструктуры (лифт, пешеходный переход, станция и т.д.), информация о которых извлекается из общедоступных

[1] https://www.openstreetmap.org/

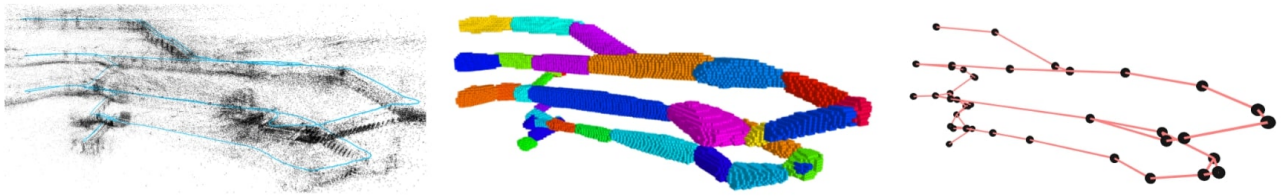

Рисунок 1.6 — (а) Разреженное облако точек, подаваемое на вход методу ТороМар; (б) выпуклые кластеры свободного пространства; (в) граф локаций, по каждой из которых строится выпуклый кластер. Источник [25].

карт. В процессе навигации по общедоступным глобальным картам и трехмерным облакам точек, приходящим с лидара робота, строится двумерная сетка занятости для каждого этажа здания и для каждого квартала города. Эксперименты, проведенные на участке города площадью 17 км^2, показали, что время планирования сокращается в среднем в три раза по сравнению со стандартной навигацией по двумерной карте, а количество потребляемой памяти сокращается до 10 раз. При этом длина построенного маршрута в сравнении с глобальной двумерной картой почти не изменилась.

В работе [27] представлен офлайн-метод построения топологической карты многоэтажного здания. На вход методу подается плотная трехмерная модель здания в виде облака точек. По облаку точек строится многоуровневая метрико-топологическая карта, состоящая из графа комнат, графа локаций, а также разбиения на двумерные и трехмерные регионы. Сначала здание разбивается на этажи, далее для каждого этажа строится воксельная сетка. Затем в воксельной сетке этажа ищутся вертикальные столбики, которые объединяются в регионы. Между соседними регионами ищутся проходы, которые становятся вершинами топологической карты. Далее по аналогии с методом ТороМар [25] регионы делятся на кластеры свободного пространства. В экспериментах, проведенных на трехмерных моделях реальных зданий, удалось достичь точности разбиения на комнаты, измеренной с помощью коэффициента корреляции Мэттью [28], равной 0.99.

В работе [29] представлен метод построения трехуровневой карты (метрической, топологической и семантической). На вход методу подаются данные с лидара и RGB-D камеры. Топологическая карта строится как граф комнат и дверей, и для каждой комнаты строится семантический граф сцены. Вершинами графа сцены являются объекты, такие как стол, кружка и т.д., а ребрами – отношения между объектами, например, «кружка стоит на столе». По данным

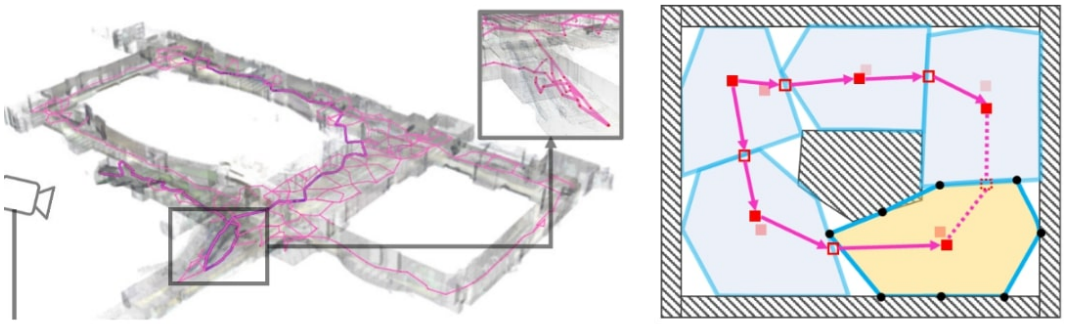

Рисунок 1.7 — (а) Скелетный граф, строящийся с помощью метода TaichiSLAM, на фоне разреженного облака точек; (б) выпуклые многогранники свободного пространства. Источник [30].

экспериментов, проведенных в симуляционной среде, время генерации графа для различных помещений составило от 15 минут до 3 часов.

В работе [30] представлен еще один офлайн-алгоритм построения топологической карты по плотному глобальному облаку точек, названный TaichiSLAM, обеспечивающий более быструю генерацию графа. Топологическая карта строится в виде скелетного графа и графа многогранников свободного пространства. Пример такой топологической карты показан на рисунке 1.7. Построение графа происходит итеративно методом поиска в ширину, на каждой итерации проводится поиск по границам между построенным многогранником и неоткартированным пространством. Вершины скелетного графа задаются как центры границ многогранников и центры самих многогранников. Время генерации графа для многоэтажного здания по данным экспериментов составило 2.8 с.

В работе [31] представлен обучаемый алгоритм Lifelong Topological Visual Navigation (LTVN), предназначенный для долгосрочной навигации в топологической карте. На вход алгоритму подается изображение с камеры робота, а также предварительно построенная топологическая карта, в каждой локации которой хранится снятое из нее изображение. Движение вдоль ребер графа локаций и проверка достижимости локаций осуществляются с помощью нейросетевых моделей. В процессе навигации ребра обновляются (стираются или добавляются) в зависимости от фактического достижения локаций. Проведенные авторами эксперименты показали успешную навигацию в симуляционных помещениях и на реальном роботе.

Онлайн-методы топологического картирования Помимо офлайн-методов топологического картирования, на настоящий момент разработано большое количество онлайн-методов, которые строят топологическую карту с нуля в реальном времени по данным с бортовых датчиков робота. Онлайн-методы подразделяются на две большие группы: методы, использующие при построении топологической карты глобальные метрические координаты (так называемые топометрические методы), и топологические методы, в которых глобальные метрические координаты не используются (но может использоваться локальная метрическая информация, например, локальная сетка занятости для каждой локации).

Топометрические методы, помимо топологической карты, как правило, предоставляют плотную метрическую карту, которая является высокодетализированным представлением окружающей среды. Использование топологических свойств среды наряду с метрическими позволяет более эффективно проводить коррекцию накапливающейся ошибки одометрии, однако за счет построения глобальной метрической карты топометрические методы зачастую ресурсозатратны, и их применение на больших расстояниях или в средах большой площади затруднительно. Так, в работе [32] представлен метод GLocal, решающий задачу исследования неизвестной местности в условиях высокой ошибки одометрии. Метод строит топологическую карту в виде графа подкарт. Каждая подкарта представляется в виде TSDF. Подкарты сшиваются в глобальную карту с помощью алгоритма Voxgraph [33], который проводит глобальную оптимизацию карты по трем группам ограничений: основанных на данных одометрии, на данных замыкания циклов и на сопоставлении пар точек на подкартах.

Метод GLocal был протестирован на симуляционных средах, представляющих собой сети тоннелей и лабиринты. В ходе экспериментов на вход методу подавались данные одометрии, в которых моделировались ошибки разной величины. Даже при самой высокой степени ошибки одометрии метод успешно завершил исследование среды, избегая столкновения с препятствиями в ходе навигации, при этом методы, основанные на классическом картировании и планировании, допустили столкновения и не смогли завершить исследование среды. Однако вычислительные затраты метода GLocal в ходе эксперимента росли пропорционально времени, прошедшему с начала исследования. Таким образом, исследование сред большой площади с помощью данного метода мо-

жет привести к нехватке вычислительных мощностей на обновление карты, и, как следствие, — к некорректному построению карты.

Одним из наиболее известных и применимых топометрических методов картирования является метод Hydra [34], разработанный в 2022 году лабораторией из Массачусетского технологического института. Метод Hydra строит трехмерный граф сцены [35], состоящий из многоуровневой топологической карты и плотной глобальной трехмерной метрико-семантической карты сцены в виде полигональной сетки. Каждый элемент этой сетки принадлежит объекту определенного типа.

Топологический граф, который строится методом Hydra, состоит из пяти слоев. Первым слоем является полигональная сетка. В вершинах этой сетки содержатся координаты, цвет, нормаль, и тип объекта. Второй слой графа описывает объекты (статические) и агенты (динамические). Вершины второго слоя содержат позицию, ограничивающий параллелепипед (далее – ОП) и тип объекта, ребра – отношения между объектами. Третий слой графа представляет собой граф локаций и структур. Локации содержат позицию и ОП без препятствий, структуры – позицию, ОП и семантический класс. Четвертый слой описывает помещения. У каждого помещения хранится позиция, ОП и тип помещения, ребрами являются двери между помещениями. Пятый слой описывает здания аналогично комнатам. Дополнительно каждый слой графа соединяется ребрами с верхним и нижним слоем. Например, каждый объект второго слоя соединен ребрами со всеми вершинами полигональной сетки, принадлежащими ему. Пример такого графа показан на рисунке 1.8.

Для построения многоуровневого топологического графа методом Hydra задействуется комплекс алгоритмов, реализующих как метрические, так и топологические подходы. Первый слой графа строится с помощью алгоритм Kimera-Semantic [36]. Третий слой графа (локации и структуры) строится с помощью алгоритма, основанного на обобщенной диаграмме Вороного [37]. Построение четвертого слоя графа (разбиение на комнаты) проводится с помощью «раздувания препятствий» (т.е. отнесения к классу препятствий всех вокселей пространства, соседствующих с вокселями препятствия) и «схлопывания дверей» (поиска компонент связности в графе локаций после нескольких итераций «раздувания препятствий»).

Замыкание циклов проходит «сверху вниз» по слоям графа. Каждой вершине в графе позиций робота задается иерархический дескриптор, который

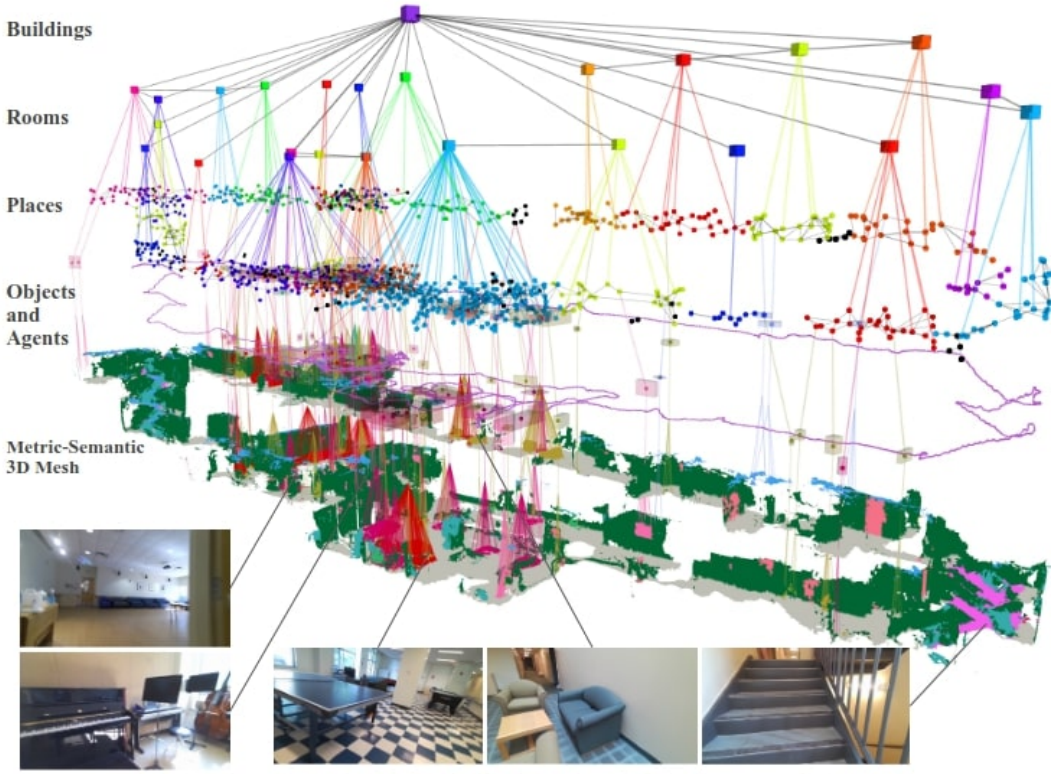

Рисунок 1.8 — Многоуровневый топологический граф, построенный по набору помещений методом Hydra. Источник [34].

состоит из классического дескриптора на базе «мешка слов» [38] и дескрипторов ближайших объектов и мест. При поиске замыкания сопоставляются сначала дескрипторы мест, затем дескрипторы объектов и визуальные дескрипторы. Подтверждение замыкания циклов и вычисление корректирующего преобразования координат происходит «снизу вверх» – сначала по графу позиций методом RANSAC [39], затем по графу объектов алгоритмом TEASER++ [40].

Метод Hydra и его обновленная динамическая версия Khronos [41] строят многоуровневую иерархическую карту, которая дает полное представление об окружающей среде, обеспечивая таким образом решение различных задач, связанных с поиском и перемещением объектов. Однако такое подробное представление окружающей среды требует высоких затрат вычислительных ресурсов и значительного времени на оптимизацию. В ходе экспериментов глобальная оптимизация графа через 20 минут работы алгоритма занимала порядка 2 с, а замыкание циклов – порядка 15 с. При этом с увеличением времени работы алгоритма увеличивались и временные затраты на обновление графа (см. рисунок 1.9). В работе [2] было показано, что при зашумленной одометрии и при больших расстояниях, пройденных роботом, метод строит несвязный граф. Таким образом, построение многоуровневой карты сред большой площади

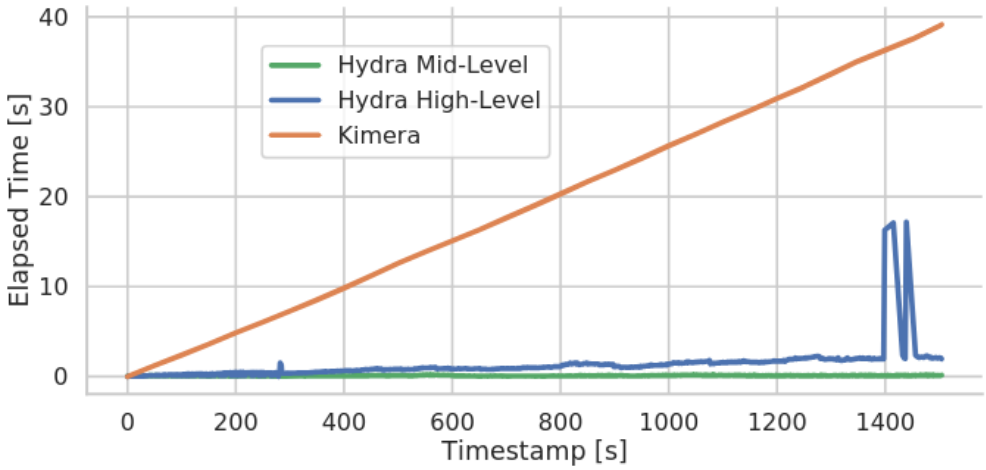

Рисунок 1.9 — График времени работы обновления карты (mid-level) и глобальной оптимизации (high-level) метода Hydra. Источник [34].

с помощью метода Hydra затруднительно и требует наличия мощных вычислителей и точных источников позиционирования.

В работе [42] был опубликован еще один метод построения многоуровневой метрико-топологической карты S-graphs+. Данный метод строит топологическую карту из четырех слоев: граф позиций робота, стен, комнат и этажей. Плоскости стен детектируются методом RANSAC. Комнаты детектируются по графу стен. Замыкание циклов осуществляется на двух уровнях: “мягкое” (по комнатам) и “жесткое” (по сопоставлению сканов). За счет исключения семантической информации и плотной полигональной сетки из графа метод S-Graphs+ показывает меньшее время обновления карты по сравнению с Hydra (порядка 200 мс против 2 с). Однако он по-прежнему подвержен накоплению ошибки одометрии и строит несвязные графы, как показано в работе [2].

Для избавления от накопления ошибки и роста вычислительных затрат, свойственных топометрическим методам, было разработано множество методов картирования, использующих только топологические свойства среды (такие, как связность локаций) без опоры на глобальные метрические координаты. Такие методы подразделяются на две группы: не использующие метрические координаты вообще и использующие локальные метрические координаты. Первая группа представлена большим разнообразием подходов, основанных на нейронных сетях и обучении с подкреплением. Принадлежность робота к локации определяется по похожести вектора признаков, извлеченного нейросетью из текущего наблюдения робота, на вектор признаков, извлеченный из локации. Движение между локациями как правило осуществляется с помощью нейросе-

ти, которая принимает на вход пару наблюдений – текущее наблюдение робота и наблюдение в целевой локации.

Большое количество нейросетевых методов топологического картирования было разработано для решения задачи однократной навигации до цели. Цель в таких задачах может быть задана либо изображением целевого объекта (Image-Goal Navigation, IGN), либо текстовой инструкцией на естественном языке (Visual-Language Navigation, VLN). Для решения задачи навигации в обеих постановках было разработано множество топологических методов. В качестве примера для задачи IGN можно привести методы NTS [43], SPTM [44], VGM [45], TSGM [46], для задачи VLN – методы из работ [47; 48]. Одним из наиболее известных методов решения задачи навигации до цели, заданной изображением, является TSGM, разработанный лабораторией обучения с подкреплением Сеульского национального университета. В качестве топологической карты метод строит граф локаций совместно с графом объектов. Каждой вершине в графе локаций приписывается нейросетевой вектор признаков панорамного изображения, полученного с этой локации.

Для локализации векторы признаков текущего изображения с робота и локации сопоставляются методом TSGM по косинусной метрике. В первую очередь текущее наблюдение робота сопоставляется с последней локализованной вершиной. Если векторы признаков не близки, то выполняется поиск вершины с близким вектором признаков по всему графу. Найденная вершина соединяется ребром с последней локализованной вершиной. Если же такая не найдена – добавляется новая вершина и соединяется ребром с последней локализованной вершиной.

С помощью TSGM была достигнута успешность навигации более 80% на коллекции симуляционных сцен Gibson [49], а также успешная навигация до цели на реальном роботе. Однако в ходе экспериментов, проведенных в работе [2], при долговременной навигации с полным объездом сцены метод TSGM соединял ребрами локации, удаленные и не связанные друг с другом в реальности (см. рисунок 1.10). Его навигационная эффективность, измеренная по метрике Success weighted by Path Length (SPL) [50], составила всего лишь 15%.

Вторая группа методов не столь многочисленна, как первая. Одним из ее представителей является метод, опубликованный в работе [51]. Метод строит комплексную топологическую карту здания, состоящую из графа свободного пространства и графа помещений. Для каждого помещения строится своя мет-

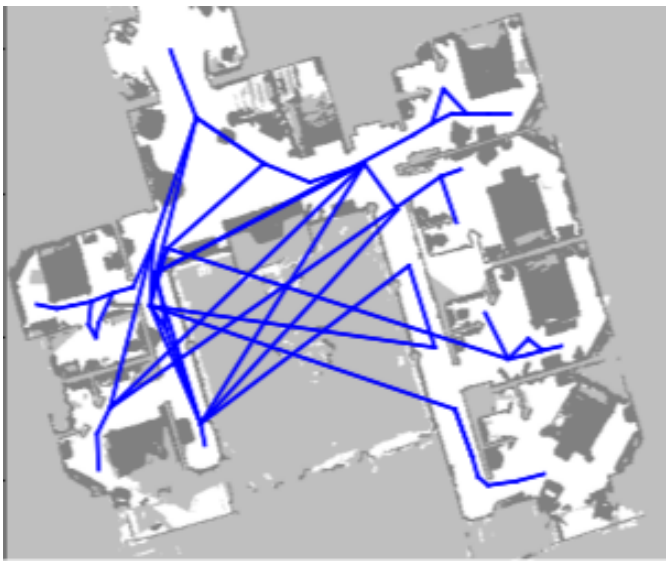

Рисунок 1.10 — Пример некорректной работы локализации в графе локаций (показан синим поверх метрической карты здания): ребрами соединены локации в разных сторонах здания.

рическая карта. Для коррекции карты используется как метрическое, так и топологическое замыкание циклов с помощью поиска изоморфизма графов. Проведенные эксперименты показали трехкратное сокращение потребляемой памяти в помещениях большой площади по сравнению с глобальной метрической картой, а также сокращение времени планирования пути более чем в 100 раз.

Другим известным топологическим методом картирования, использующим локальную метрическую информацию, является TLF [52]. Метод строит граф локаций, проводит глобальную локализацию в графе и привязку к текущей вершине с обновлением относительной позиции по одометрии. Каждой локации приписывается своя локальная метрическая подкарта. На ребрах записываются относительные позиции, а также опционально проходимость ребра. В методе TLF имеется механизм учета изменений среды (время суток, погода, динамические объекты и т.д.). Этот механизм добавляет новые вершины, если робот из-за изменений среды не может локализоваться, а затем переключается обратно на локализацию при ее восстановлении. При навигации предпочтительно используются вершины, добавленные в похожее время суток на текущее.

Эксперименты проводились на улице на колесном роботе, навигация осуществлялась по стереокамере. В ходе экспериментов один и тот же километровый маршрут был пройден 21 раз в разное время суток при наличии динамических объектов. Среднее время локализации на всех прогонах соста-

Таблица 1 — Свойства рассмотренных методов картирования

Методы		Онлайн	Выч. эфф.	Работа на большой площади	Работа без накопления ошибки	Работа внутри и вне помещений	Открытая реализация
Метрические	RTAB-Map [16]	✓	✗	✗	✗	✓	✓
	Voxblox [21]	✓	✗	✗	✗	✓	✓
Топологические офлайн	TopoMap [25]	✗	✓	✓	✓	✗	✗
	City-scale [26]	✗	✓	✓	✓	✓	✗
	Hierarchical [27]	✗	✓	✓	✓	✗	✗
	LTVN [31]	✗	✓	✗	✓	✗	✗
Топометрические	GLocal [32]	✓	✗	✗	✓	✓	✓
	Hydra [34]	✓	✗	✗	✗	✗	✓
	S-graphs+ [42]	✓	✗	✗	✗	✗	✓
Топологические без координат	SPTM [44]	✓	✓	✗	✓	✗	✓
	VGM [45]	✓	✓	✗	✓	✗	✓
	TSGM [46]	✓	✓	✗	✓	✗	✓
	ETPNav [48]	✓	✓	✗	✓	✗	✓
	CMTP [47]	✓	✓	✗	✓	✗	✗
Топологические с локальными координатами	Gomez et al. [51]	✓	✓	✓	✓	✗	✗
	TLF [52]	✓	✓	✓	✓	✗	✗

вило порядка 100 мс. Средняя ошибка локализации составила порядка 0.5 м и 4 градуса. Таким образом, метод TLF обеспечивает надежную долговременную навигацию на открытых пространствах, однако он не предназначен для работы внутри помещений и не может использовать лидарные данные для более точного позиционирования. Метод не имеет открытой реализации в виде комплекса программ, что затрудняет его применение на практике и валидацию описанных авторами результатов.

Обобщая приведенные выше описания методов и алгоритмов картирования, можно выделить ряд свойств, важных для использования методов в навигации робота и присущих тем или иным методам или классам методов:

1. Построение карты в реальном времени (онлайн);
2. Вычислительная эффективность – низкое потребление памяти и малое время обновления карты по наблюдениям с робота в течение всего периода работы алгоритма;
3. Применимость метода для картирования сред большой площади (работа на большой площади);
4. Устранение накопления ошибки позиционирования в ходе работы алгоритма (работа без накопления ошибки);
5. Применимость метода для картирования помещений и открытых сред;

6. Наличие в открытом доступе реализации метода в виде комплекса программных средств.

Характеристика рассмотренных методов с использованием указанных свойств представлена в таблице 1. Ни один из рассмотренных методов не обладает всеми перечисленными свойствами и не может обеспечить точное картирование закрытых и открытых сред большой площади в реальном времени. Топологический метод TLF, использующий локальные метрические координаты, обладает вычислительной эффективностью и применим для картирования сред большой площади в реальном времени, однако не предназначен для работы в помещениях и не имеет открытой реализации. В данной работе разрабатывается новый вычислительно эффективный алгоритм топологического картирования с использованием локальных метрических координат, пригодный для картирования больших площадей как внутри, так и вне помещений.

1.3 Методы и алгоритмы локализации

Современные методы визуальной и лидарной одометрии позволяют отслеживать местоположения робота с высокой относительной точностью (порядка 0.5%). На траекториях небольшой длины (например, в пределах коридора здания) такие ошибки будут незначительны, однако на расстоянии в несколько километров ошибка может уже достигать десятков метров, что приведет к невозможности локализации и планирования маршрутов. Таким образом, возникает необходимость в корректировании ошибки путем периодической глобальной привязки к внешним ориентирам, положение которых на карте известно. На улице накопившуюся ошибку можно корректировать до нескольких метров путем позиционирования по ГНСС, однако во многих задачах (например, движение беспилотного автомобиля по городу) такой точности бывает недостаточно. Необходимость привязки к ориентирам возникает внутри зданий и в подземных средах, где точное позиционирование по ГНСС недоступно и требуется высокая точность определения местоположения. Так возникает задача глобальной локализации.

Современные методы локализации по карте делятся на три большие группы. Первая группа методов сопоставляет текущее наблюдение с робота

(представляемое в виде изображения или облака точек) сразу со всей картой. Такие методы могут использоваться для локализации в метрической карте или в некоторой подкарте топологической карты. Вторая группа методов, называемая методами распознавания места (в англоязычной литературе – Place Recognition), использует для локализации базу изображений и/или облаков точек. Локализация выполняется путем поиска в базе элемента, наиболее похожего по некоторым признакам на текущее наблюдение с робота. В качестве меры похожести может выступать евклидово или косинусное расстояние между дескрипторами (векторами признаков, кодирующими наблюдения). Такой подход может применяться для локализации в топологической карте, представленной в виде графа локаций – по графу создается база, в которую добавляются наблюдения с каждой локации, и текущая локация определяется путем поиска наиболее похожего наблюдения из базы. Третья группа методов сочетает в себе преимущества первых двух групп. Представленные в ней методы сначала ищут примерное положение (локацию) в глобальной карте с помощью технологий распознавания места, а затем сопоставляют текущее наблюдение с робота с картой окрестности найденного примерного положения, находя таким образом точное положение в карте. Так достигается точное определение местоположения робота в топологических картах и в метрических картах большого размера. Все три группы методов подробно рассмотрены ниже.

Методы сопоставления текущего наблюдения с глобальной картой Методы автоматической глобальной локализации в карте начали развиваться в 1990-х годах. Преимущественно это были метрические методы, в которых карта представлялась в виде облака точек. Прорывным методом глобальной локализации на карте по лидарным данным стал метод локализации Монте-Карло [53], разработанный в 1999 году исследователями из Питтсбурга. В этом методе глобальная локализация проводится путем сопоставления облака точек с лидара и глобального облака точек, задающего карту. Для сопоставления вместо теоретических вероятностных подходов используется семплирование методом Монте-Карло. С помощью такого подхода робот успешно локализовался внутри музея на траектории длиной 2 км без накопления ошибки.

Одним из наиболее распространенных современных методов глобальной локализации по карте является IRIS [54]. Метод использует геометрическое сопоставление особых точек на данных с камеры робота и точек глобальной

карты. Сопоставление производится вероятностными методами и методами оптимизации. В среде масштаба порядка сотен метров с точной картой такой подход позволяет достичь точности локализации порядка 0.5 м при использовании только данных с камеры.

В работе [55] облака точек с лидара проецируются на плоскость в двуслойную локальную карту. Ячейки первого слоя кодируют бордюры, детектируемые авторским алгоритмом. Ячейки второго слоя кодируют все точки облака, находящиеся над плоскостью дороги. На этапе локализации локальная карта сопоставляется с глобальной двуслойной картой с помощью метода Монте-Карло с байесовской фильтрацией. Такой подход позволил получить точность локализации порядка 8 см на расстоянии в 1.3 км при локализации движущегося по дороге автомобиля. В работе [56] представлен алгоритм глобальной локализации автомобиля на карте с помощью метода Монте-Карло и картирования вертикальных структур, таких как фонарные столбы и деревья. Эксперименты, проведенные на маршруте длиной в 10 км, подтвердили высокую точность локализации с помощью представленного алгоритма – средняя ошибка составила 10 см и 0.18°.

Методы распознавания места Помимо методов локализации при помощи сопоставления текущего наблюдения с глобальной метрической картой, в 2000-х годах появились методы локализации с помощью поиска в базе изображений и/или облаков точек элемента, наиболее похожего на текущее наблюдение с робота. Такие методы в академической литературе называются методами распознавания места (англ. Place Recognition). Первые методы распознавания мест представляют изображение в виде «мешка слов» (англ. Bag of Words) – каждое изображение с камеры или облако точек представляется в виде набора дескрипторов. Результатом локализации является позиция изображения из базы, наиболее похожего на текущее изображение с робота с точки зрения близости дескрипторов. Одним из первых методов, использующих эту концепцию, стал FAB-MAP [57]. В нем из изображения извлекаются особые точки, по каждой из которых вычисляется дескриптор алгоритмом SURF [58]. По полученному набору дескрипторов ищется наиболее похожее изображение в базе для поиска уже посещенных роботом мест и коррекции карты (замыкания циклов). Одним из наиболее известных методов, использующих концепцию «мешка слов», стал метод DBoW2 [38], в котором изображения представляются в виде набора

бинарных дескрипторов. Метод был разработан в 2012 году исследователями из Сарагосы и используется для замыкания циклов в современных алгоритмах ОКЛ, таких как ORB-SLAM3 [10].

Помимо методов, основанных на дескрипторах, извлекаемых классическими методами, в последнее время было разработано множество нейросетевых методов распознавания места. Одним из первых таких методов является NetVLAD [59]. По каждому изображению с камеры генерируется векторное представление (дескриптор) с помощью сверточной нейронной сети с особым слоем – вектором локально агрегированных дескрипторов. Локализация происходит путем сопоставления дескриптора текущего изображения с робота и дескрипторов изображений из базы. Такой подход позволил достичь точности распознавания места порядка 70% и полноты порядка 80-85% на наборе данных Pitts250k-test [60]. При этом использование слоя VLAD и тройной функции потерь (англ. triplet loss) позволило достичь успешного распознавания мест, которые в базе изображений были сняты с другого ракурса и/или в другое время суток, чем текущее наблюдение с робота.

После NetVLAD появилось множество других методов распознавания места по изображению. Так, в 2022 году вышел метод CosPlace [61], в котором задача распознавания места представляется как задача классификации, и классы изображений разделены географически. Такой подход позволил значительно уменьшить размерность дескриптора при сохранении точности распознавания места, что привело к значительной экономии памяти при использовании на больших базах изображений. В 2023 году вышел метод распознавания места MixVPR [62], основанный на трансформерной архитектуре нейронной сети. Использование трансформера позволило достичь большей устойчивости к смене ракурса, времени суток и погодных условий, чем у методов, основанных на сверточных нейронных сетях (см. рисунок 1.11). Полнота распознавания места на наборе данных Pitts250k-test составила 88%.

Изображения содержат двумерную информацию об изображенной местности. Снимки одного и того же места значительно различаются в разное время суток, при съемке с разных ракурсов и при разных погодных условиях. Все это создает дополнительные сложности при разработке методов распознавания места. Альтернативным способом представления участка местности является облако точек, которое может быть снято с лидара робота – оно содержит более полную трехмерную информацию о сцене и менее подвержено изменениям при

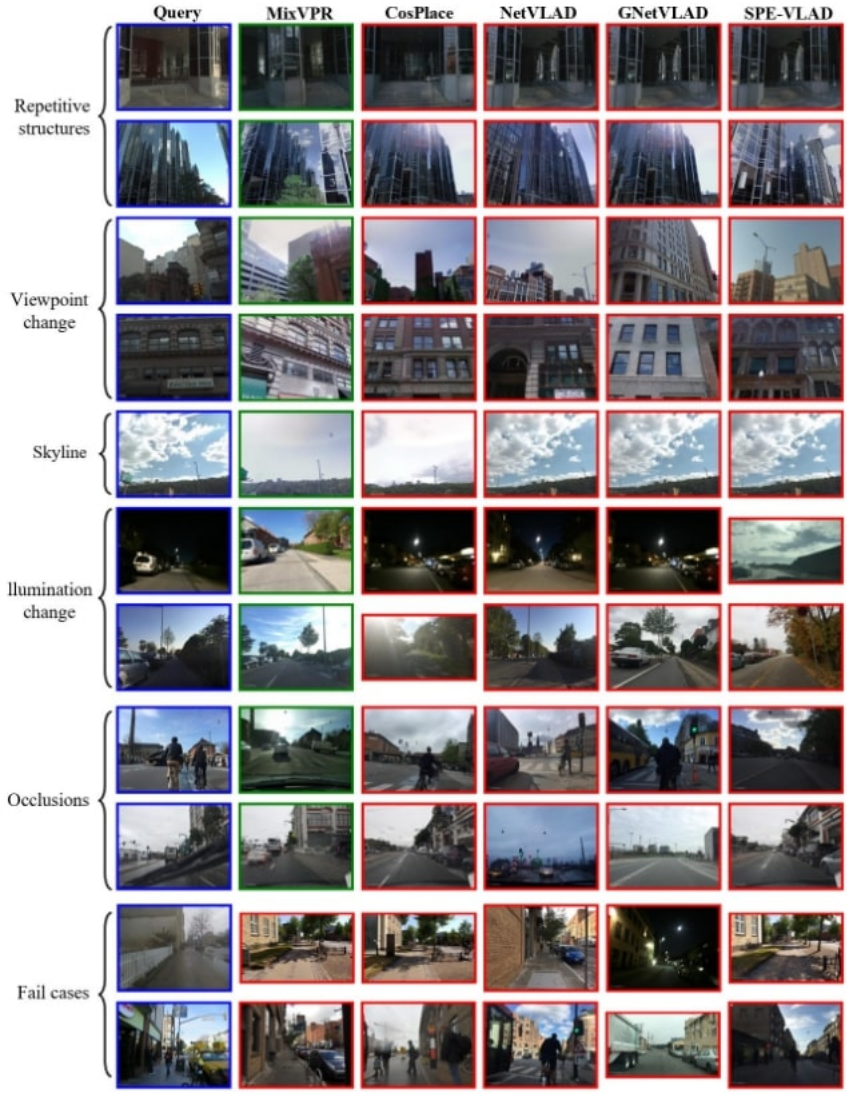

Рисунок 1.11 — Примеры успешного и неудачного распознавания места различными нейросетевыми методами. Источник [62].

смене времени суток и погодных условий. В настоящее время разработано множество методов нейросетевого распознавания места по облакам точек. Одним из первых таких методов является PointNetVLAD [63], основанный на нейронной сети PointNet [64] и векторе локально агрегированных дескрипторов VLAD. На вход нейросети подается облако точек в исходном виде, на выходе из облака извлекается дескриптор размерности от 128 до 512. Такой подход позволил достичь полноты распознавания места 80-90% на различных наборах данных.

В методе MinkLoc3d [65] представлен другой подход к распознаванию места по облакам точек, основанный на пространственной дискретизации облака и сверточной нейросетевой архитектуре. Облако точек представляется в виде трехмерной сетки определенной размерности, в каждой ячейке которой содержится информация о присутствии или отсутствии точек облака. Полу-

ченная сетка подается на вход трехмерной сверточной нейросети, которая генерирует дескриптор размерности 256. Такой подход позволил достичь полноты распознавания места 88-98% при сохранении скорости, достаточной для работы в реальном времени (десятки миллисекунд на обычном графическом процессоре). Метод SVT-Net [66] позволил достичь еще более высокой полноты распознавания места (90-98%) при сохранении высокой скорости работы за счет использования трансформерной архитектуры и механизма внимания.

Методы распознавания места по облакам точек в среднем позволяют добиться более высокого качества распознавания, однако в некоторых ситуациях могут работать хуже, чем визуальные методы. Так, облака точек имеют ограниченный размер (как правило, десятки метров), в то время как на изображения попадают ориентиры, расположенные за сотни метров или за километры от точки съемки. Качество распознавания места по лидарным облакам точек может понижаться во время дождя или снега из-за физических особенностей лидаров. Так, по данным из работы [65], точность распознавания места на наборе данных RobotCar Seasons [67] составила 66% на сценах, записанных во время дождя, и 87% на сценах, записанных в солнечную погоду. Для решения подобных проблем могут применяться мультимодальные методы распознавания места, которые принимают на вход как облако точек с лидара, так и изображения с камер.

В работе [68] представлен мультимодальный метод, в котором облако точек обрабатывается нейросетевой архитектурой из метода PointNetVLAD, а изображение – общеизвестной сверточной нейронной сетью ResNet-50 [69]. Векторы признаков, предсказанные по облаку точек и по изображению, обрабатываются еще одним полносвязным слоем для генерации совместного дескриптора места. Такой подход позволил достичь полноты в 88-98% на наборе данных KITTI [70], содержащем облака точек и изображения, записанные при проезде автомобиля по городу. Такое качество было достигнуто за счет взаимного дополнения признаков, извлекаемых из облаков точек и изображений (см. рисунок 1.12).

В работе [71] представлено мультимодальное расширение метода MinkLoc3D, названное MinkLoc++. Архитектура нейросети для распознавания места состоит из двух веток – одна генерирует дескриптор облака точек, вторая генерирует дескриптор изображения. Схема архитектуры представлена на рисунке 1.13. Итоговый дескриптор места получается конкатенацией

Рисунок 1.12 — Взаимное дополнение информации, содержащейся в облаках точек и в изображениях. Источник [68].

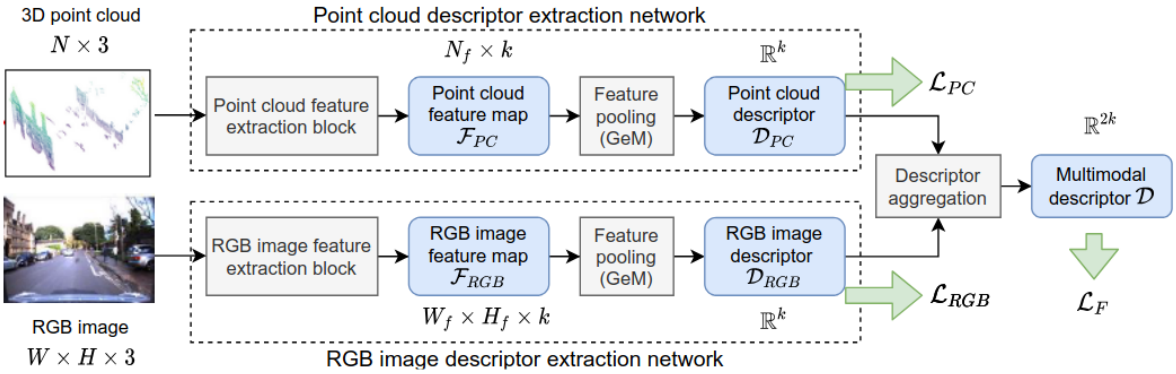

Рисунок 1.13 — Взаимное дополнение информации, содержащейся в облаках точек и в изображениях. Источник [71].

дескрипторов облака точек и изображения. Для обучения нейросети использовалась тройная функция потерь – комбинация функции потерь на дескрипторах облака точек и изображения по отдельности, а также функции потерь на совместном дескрипторе. Такой подход позволил достичь полноты распознавания места 97% на наборе данных KITTI по сравнению с 94% при использовании только облака точек. В работе [72] представлено расширение метода MinkLoc++, позволяющее использовать вместо одного изображения пару изображений, снятых камерами робота в противоположных направлениях, а также дополнительно использовать текстовую информацию и данные семантической сегментации. С помощью такого подхода удалось достичь еще более высокой полноты распознавания места, равной 98%.

Именно методы, основанные на распознавании мест, наиболее применимы для локализации в топологической карте, представленной в виде графа локаций, поскольку они позволяют быстро определить локацию, в которой на-

ходится робот. Однако ложноположительные результаты распознавания могут привести к соединению внешне похожих, но удаленных друг от друга локаций при построении графа, что может привести к сбоям навигации. В качестве примера можно привести некорректную работу метода топологического картирования TSGM [46], описанную в работе [2]. За счет ложноположительных результатов нейросетевого распознавания мест соединялись ребрами локации, расположенные в двух различных, но внешне похожих друг на друга коридорах здания, как показано на рисунке 1.10. Таким образом, для корректной локализации в графе необходима дополнительная фильтрация распознанных мест, которая может быть выполнена с помощью сопоставления сканов в комбинированных методах глобальной локализации.

Комбинированные методы глобальной локализации определяют местоположение робота в глобальной карте в два этапа. Вначале ищется приблизительное положение робота в карте с помощью методов распознавания места, далее в его окрестности ищется точное положение робота в карте путем сопоставления текущего наблюдения с робота с соответствующим участком карты или сканом. Таким образом, за счет сужения зоны для сопоставления текущего наблюдения с робота можно найти точное положение робота даже в картах большой размерности с низкими вычислительными затратами. К примеру, если карта хранится в виде набора локаций, и каждая локация описывается сканом с лидара робота, то задача сопоставления наблюдения с картой сводится к задаче сопоставления двух сканов — текущего скана робота и скана найденной локации.

Одним из наиболее известных и применимых методов сопоставления лидарных сканов является итеративный метод ближайшей точки (англ. ICP) [73]. Метод сопоставляет два облака точек последовательно, начиная с некоторого начального приближения. На каждом шаге к каждой точке первого облака ставится в соответствие ближайшая к ней точка второго облака. Далее методом наименьших квадратов ищется преобразование координат, минимизирующее суммарное квадратичное отклонение между парами сопоставленных точек. Найденное преобразование применяется к первому облаку, и процесс повторяется. Для ускорения метода вместо всех точек облака могут браться только особые точки, вычисленные, например, методами из работ [74], [75].

Для достижения хорошего качества работы метода ICP необходимо задать начальное приближение преобразования координат, сопоставляющего два скана. Для поиска такого приближения может использоваться метод RANSAC [39] (англ. RANdom SAmple Consensus). Это итеративный метод для поиска наилучших параметров модели. В случае с поиском начального сопоставления для двух облаков точек, сначала по каждому облаку вычисляются особые точки, затем между особыми точками строятся соответствия. На каждой итерации выбирается случайная подвыборка соответствий, и по ней вычисляется оптимальное преобразование координат и оценивается его качество. Соответствия, не подходящие под найденное преобразование, помечаются как выбросы и удаляются. В качестве ответа выбирается преобразование с наилучшим качеством сопоставления среди всех итераций. Так можно решать задачу сопоставления сканов, однако для успешного поиска сопоставлений облака точек должны иметь значительную долю перекрытия, как показано в работе [4]. При локализации в топологической карте значение перекрытия облаков точек при попытке сопоставления часто бывает в диапазоне от 25 до 50%, а при таком перекрытии значение полноты сопоставления сканов составило всего 53%. Среднее время работы методов RANSAC + ICP, измеренное в работе [4], составило 360 мс.

Для повышения быстродействия сопоставления сканов, а также для стабильной работы в условиях низкой доли перекрытия сканов применяют алгоритмы, основанные на двумерных проекциях облаков точек. Так, в алгоритме BVMatch [76] используется проекция облака точек на горизонтальную плоскость в виде карты плотностей участков плоскости. По карте плотностей с помощью нейронной сети вычисляется локальный дескриптор, далее по локальным дескрипторам вычисляется преобразование координат между двумя облаками точек с помощью метода RANSAC. Алгоритм BVMatch был объединен с методом распознавания места в состав комбинированного метода глобальной локализации BEVPlace++ [77]. С помощью комбинированного подхода авторы метода BEVPlace++ достигли точности локализации порядка 40 см на наборах данных KITTI [70] и NCLT [78]. Однако результат был измерен авторами только на точках траектории с корректно распознанным местом – в случае некорректного распознавания места метрическая ошибка не вычислялась.

В методе BoxGraph [79] для распознавания места применяется семантический граф сцены, а локальное уточнение позиции проводится с помощью сингулярного разложения и метода RANSAC. Такой комбинированный подход

Таблица 2 — Свойства рассмотренных методов локализации

Методы		Скорость	Точность	Компактность карты	Работа на большой площади	Открытая реализация
Сопоставление наблюдения с картой	IRIS [54]	✓	✓	✗	✓	✓
	Wang et al. [55]	✗	✓	✗	✓	✗
	Li et al. [56]	✗	✓	✗	✓	✗
Распознавание места	DBoW2 [38]	✓	✗	✓	✓	✓
	NetVLAD [59]	✓	✗	✓	✓	✓
	CosPlace [61]	✓	✗	✓	✓	✓
	MixVPR [62]	✓	✗	✓	✓	✓
	PointNetVLAD [63]	✗	✗	✓	✓	✓
	MinkLoc3D [65]	✓	✗	✓	✓	✓
	SVT-Net [66]	✓	✗	✓	✓	✓
	MinkLoc++ [71]	✓	✗	✓	✓	✓
Комбинированные	Расп. места + ICP [73]	✗	✗	✓	✓	✓
	Расп. места + RANSAC [74] + ICP	✗	✓	✓	✓	✓
	BEVPlace++ [77]	✗	✓	✓	✓	✓
	BoxGraph [79]	✗	✓	✓	✓	✓
	GLFP [80]	✓	✓	✓	✗	✗

позволил достичь практически 100% полноты и точности 88% на наборе данных KITTI, наряду с ошибкой локализации порядка 8 см. При этом максимальная ошибка локализации не превысила 50 см. Однако метод BoxGraph требует решения задачи семантической сегментации облака точек, что, в свою очередь, требует высоких затрат вычислительных ресурсов. Данные о быстродействии метода авторами не приведены.

Для устранения ошибок локализации, возникающих вследствие ложноположительных результатов распознавания места, может использоваться фильтрация результатов на основе информации о локальном перемещении робота (одометрии). Например, в методе GLFP [80] данные одометрии используются для определения позиции робота в помещении между выполнениями итераций глобальной локализации. Данный метод принимает на вход вместо плотной глобальной карты план помещения и при этом показывает точность локализации, сопоставимую с методами локализации по глобальной лидарной карте. Однако метод не предназначен для работы в открытых пространствах и помещениях большой площади, а также не имеет открытой реализации в виде комплекса программных средств.

Характеристика рассмотренных методов, алгоритмов и моделей локализации представлена в таблице 2. Приведены четыре свойства, важные для эффективного использования методов локализации при навигации роботов:

1. Скорость, достаточная для работы метода в реальном времени (100-200 мс на обработку одного наблюдения с робота);
2. Точность локализации, достаточная для корректного позиционирования робота и корректного планирования пути (порядка 10 см внутри помещений и 0.5 м в открытых средах);
3. Малый объем памяти, занимаемой картой, по которой проводится локализация (компактность карты);
4. Применимость метода для локализации в средах большой площади;
5. Наличие открытой реализации метода в виде комплекса программ.

Методы сопоставления наблюдения с робота с глобальной метрической картой (в частности, IRIS [54]) могут обладать высокой скоростью и точностью работы, однако требуют наличия плотной метрической карты, которая при работе в средах большой площади занимает значительные объемы памяти. Методы распознавания места используют для локализации базу дескрипторов небольшой размерности, извлеченных из наблюдений с робота, однако не дают требуемой точности, поскольку выдают в качестве результата позицию ближайшего дескриптора и подвержены ошибкам распознавания. Для решения проблемы низкой точности применяется комбинация методов распознавания места и сопоставления наблюдений (сканов). Однако большинство комбинированных методов не обладает скоростью, достаточной для эффективной работы в реальном времени. В данной работе разрабатывается вычислительно эффективный комбинированный метод локализации, обладающий достаточной скоростью для работы в реальном времени и приемлемой точностью локализации.

1.4 Выводы по главе

Построение карты местности и автономная локализация в построенной карте являются одними из ключевых задач для обеспечения автономной навигации робототехнической системы в неизвестной среде. Для восприятия и

картирования окружающего пространства существуют различные типы бортовых датчиков, наиболее распространёнными из которых являются лазерные сканеры (лидары) и камеры. Помимо наблюдений с лидаров и камер, для картирования и локализации используется одометрия – оценка перемещения робота в пространстве по инерциальным или колесным датчикам, либо по датчикам восприятия. Задача картирования заключается в создании модели местности (карты) по наблюдениям с датчиков и данным одометрии. Задача локализации заключается в определении положения робота по наблюдениям с датчиков, данным одометрии и построенной карте.

Алгоритмы картирования подразделяются на метрические (карта представляется в виде геометрических структур, например, сеток) и топологические (карта представляется в виде графа локаций). Также распространены топометрические алгоритмы, строящие метрическую и топологическую карту совместно. Основными недостатками метрических и топометрических алгоритмов являются высокие затраты вычислительных ресурсов и памяти, а также накопление ошибки позиционирования и долгое планирование пути по построенной карте. Топологические алгоритмы позволяют устранить эти недостатки, однако на настоящий момент универсального и надежного алгоритма топологического картирования, работающего на лидарных данных на больших расстояниях, в литературе не представлено.

Алгоритмы локализации в карте подразделяются на три группы: сопоставление текущего наблюдения с единой глобальной картой, поиск подходящей локации в карте с помощью методов распознавания места, а также комбинацию первых двух подходов – нахождение точного положения робота в карте путем последовательного распознавания места и сопоставления текущего наблюдения с найденной локацией. Использование первого подхода на картах большого размера приводит к значительным временным затратам. Методы распознавания места позволяют быстро находить текущую локацию в графах большого размера, однако выдают ложноположительные результаты, приводящие к сбоям локализации. Комбинированные методы позволяют решить данную проблему, однако для точной локализации необходима комплексная фильтрация ошибок, основанная на сопоставлении сканов и оценке перемещения робота.

На основании вышеизложенного можно сделать вывод о том, что на настоящий момент не разработаны универсальные алгоритмы картирования и алгоритма локализации, подходящие для работы в открытых и закрытых сре-

дах большой площади, вычислительно эффективные и имеющие открытую реализацию. При этом наиболее перспективными для создания таких алгоритмов является топологический подход к картированию с использованием локальных метрических координат, а также комбинированный подход к локализации, сочетающий в себе распознавание места и сопоставление текущего наблюдения с окрестностью распознанного места. Проблемы существующих методов и алгоритмов и отсутствие универсального алгоритма ОКЛ определяют задачи диссертационной работы:

1. Провести анализ существующих методов топологического картирования и локализации.
2. Построить математическую модель задачи топологического картирования и локализации и оценки качества ее решения.
3. Разработать вычислительно эффективный алгоритм топологического картирования и локализации, обладающий высоким качеством локализации и построенной карты.
4. Создать программный комплекс топологического картирования и локализации, провести экспериментальные исследования разработанных алгоритмов с использованием предложенной математической модели в симуляционных средах и на реальных робототехнических системах.

Глава 2. Постановка задачи топологического картирования и локализации

Оценка качества алгоритмов ОКЛ является важной задачей, решение которой необходимо для сравнения алгоритмов между собой и выбора наиболее применимого алгоритма для решения задачи ОКЛ в той или иной среде. Для оценки качества метрических методов ОКЛ, как правило, используется абсолютная и относительная ошибка траектории, вычисляемая по расхождению между истинной траекторией робота и результатами локализации. Оценка качества топологических методов ОКЛ путем сравнения их результатов с эталонными данными затруднительна, т.к. топологические карты часто не содержат глобальных метрических координат.

В данной главе представлен подход к оценке качества картирования и локализации, учитывающий особенности топологических карт. Описана математическая модель топологического картирования (построения графа локаций) и локализации в топологической карте по облакам точек и одометрии, полученным с наблюдений с бортовых датчиков робота. Предложена адаптация стандартных критериев качества локализации (абсолютная и относительная ошибка траектории) для топологических карт, учитывающая особенности локализации в графе локаций.

Для оценки качества графов локаций предложен способ, основанный на вычислении эффективности путей, проложенных с помощью оцениваемого графа между различными парами точек. Предложенный способ подходит для оценки качества как топологических, так и метрических карт (в случае метрической карты графом может являться сетка занятости, в которой соседние свободные ячейки соединены ребрами). Предложенный критерий отражает эффективность применения графа локацй для планирования путей при навигации мобильного робота. Помимо путевой эффективности, предложен ряд других параметров графа локаций, влияющих напрямую на его применимость для навигации мобильного робота: количество компонент связности, доля покрытия среды локациями графа, корректность ребер и полнота графа.

2.1 Математическая модель окружающей среды и наблюдений

Предполагается, что робот оснащен программно-аппаратным источником одометрии (инерциальной, колесной, визуальной, лидарной или комплексированной) и трехмерным панорамным датчиком восприятия (панорамная камера глубины или лидар). Таким образом, наблюдения с датчиков робота преобразуются в облако точек и одометрию, являющиеся входными данными в задаче картирования и локализации. В процессе картирования робот движется по некоторой предварительно заданной дискретной траектории и получает наблюдения в каждой ее точке.

В рамках данной модели робот движется в трехмерном евклидовом пространстве. Используются две системы координат: глобальная, связанная с окружающей средой, и связанная с роботом. В качестве начала координат во второй системе на роботе выбирается некоторая точка, например, точка фокуса его камеры. В качестве базиса системы координат робота на нем выбираются три взаимно перпендикулярных направления (как правило, одним из таких направлений является главная оптическая ось камеры робота или ось его ведущих колес). Положение и ориентация робота в глобальной системе координат определяются по выбранной точке на роботе и выбранному на нем базису. По положению и ориентации задается преобразование координат из глобальной системы в систему координат робота. Формальные определения даны ниже.

Определение 2.1. Окружающая среда $W \subset \mathbb{R}^3$ представляется в виде ограниченной замкнутой области трехмерного евклидова пространства с выбранной на нем правой прямоугольной декартовой системой координат. Она делится на замкнутое связное **множество объектов** W_{obj} и **свободное пространство** W_{free}:

$$W = W_{free} \cup W_{obj} (W_{free} \cap W_{obj} = \varnothing). \tag{2.1}$$

Для навигации наземных роботов в среде с горизонтальной поверхностью пола вместо трехмерной окружающей среды может рассматриваться срез окружающей среды $W_{z_1:z_2}$, охватывающий высоты от z_1 (максимальная высота барьера, преодолимого роботом, всё, что находится выше, является для робота

препятствием) до z_2 (высота верхней части робота). Тогда окружающая среда будет плоской:

$$\begin{aligned} W_{z_1:z_2} &= W_{free}^{z_1:z_2} \cap W_{obj}^{z_1:z_2}; \\ W_{obj}^{z_1:z_2} &= (x,y) \in \mathbb{R}^2 : \exists z \in [z_1; z_2] \; s.t. \; (x,y,z) \in W_{obj}; \\ W_{free}^{z_1:z_2} &= (x,y) \in \mathbb{R}^2 : \forall z \in [z_1; z_2] \; (x,y,z) \in W_{free}. \end{aligned} \tag{2.2}$$

Определение 2.2. Позицией робота $\mathbf{p} = (p^x, p^y, p^z) \in \mathbb{R}^3$ назовем координаты выбранной на нем точки.

Определение 2.3. Ориентацией робота $Q \in SO(3)$ назовем вращение, переводящее базис глобальной системы координат в базис системы координат робота. Здесь и далее $SO(3)$ обозначает матричную группу вращений вокруг начала координат в пространстве $\mathbb{R}^3$.

Определение 2.4. Позой робота $s = (\mathbf{p}, Q)$ назовем кортеж, состоящий из его позиции и ориентации.

Траекторию движения робота обозначим как $P = (s_0, s_1, \ldots, s_{t_{max}})$. Траектория состоит из поз робота $s_t = (\mathbf{p_t}, Q_t)$ в каждый момент t:

$$s_t = (\mathbf{p_t}, Q_t); \; \mathbf{p_t} = (p_t^x, p_t^y, p_t^z) \in \mathbb{R}^3; \; Q_t \in SO(3). \tag{2.3}$$

Каждой позе s_t соответствует аффинное преобразование S_t, переводящее из системы координат робота в глобальную систему координат:

$$S_t = \left(\begin{array}{ccc|c} & Q_t & & \begin{matrix} p_t^x \\ p_t^y \\ p_t^z \end{matrix} \\ \hline 0 & 0 & 0 & 1 \end{array}\right) \in SE(3), \tag{2.4}$$

здесь и далее $SE(3)$ обозначает матричную группу однородных преобразований в трёхмерном пространстве, состоящих из перевода и вращения для правой декартовой системы координат.

Применение такого преобразования к трехмерной точке $\mathbf{p} = (x, y, z)^T \in \mathbb{R}^3$ выглядит следующим образом:

$$S_t \cdot \mathbf{p} = \left(\begin{array}{ccc|c} & Q_t & & \begin{matrix} p_t^x \\ p_t^y \\ p_t^z \end{matrix} \\ \hline 0 & 0 & 0 & 1 \end{array}\right) \cdot \begin{pmatrix} x \\ y \\ z \\ 1 \end{pmatrix}. \tag{2.5}$$

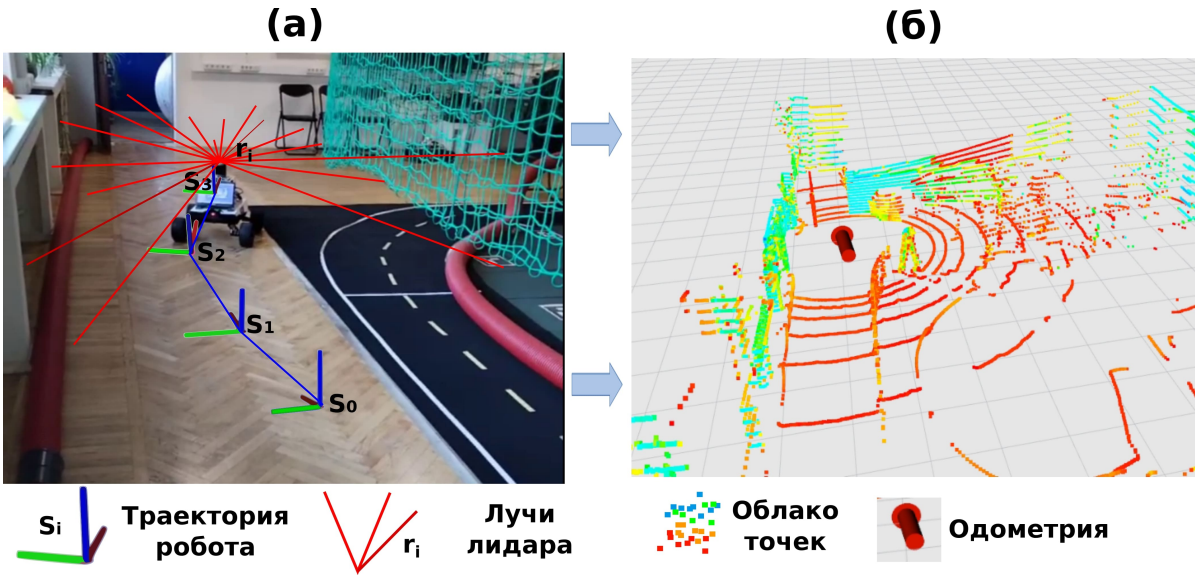

Рисунок 2.1 — Пример получения входных данных для задачи ОКЛ с робота: (а) траектория и наблюдение с датчика восприятия; (б) облако точек и одометрия.

Последовательное применение (композиция) преобразований и обратная композиция (разность) преобразований обозначаются символами $\oplus$ и $\ominus$ соответственно.

Наблюдение окружающей среды из позы s_t состоит из показаний датчика восприятия D_t и показаний датчиков, используемых для оценки перемещения робота (такими датчиками могут быть энкодеры колес, инерциальные датчики, камеры и лидары). По наблюдениям вычисляются облако точек и одометрия, используемые в качестве входа для задачи ОКЛ. Пример получения такого облака точек показан на рисунке 2.1. D_t представляет собой список расстояний до объектов среды, измеренных лидаром или RGB-D камерой с некоторой погрешностью:

$$\begin{aligned} D_t &= (D_t^1, \ldots, D_t^n); \\ A &= \{\alpha : \ S_t \cdot \alpha \mathbf{r_i} \in W_{obj}; \alpha < \alpha_{max}\}; \\ D_t^i &= \begin{cases} \min A + \varepsilon_t^i, & A \neq \varnothing \\ 0, & A = \varnothing \end{cases}; \\ \varepsilon_t^i &\sim \mathcal{N}(0, \sigma^2). \end{aligned} \tag{2.6}$$

Вектор $\mathbf{r_i}$ представляет собой вектор направления i-го луча лидара или i-го пикселя RGB-D камеры в системе координат робота, имеющий единичную длину. Значение α_{max} является предельным расстоянием измерения датчика восприятия. Множество A включает в себя расстояния, на которых луч, направленный

вдоль вектора $\mathbf{r_i}$, пересек точки объектов среды – измерением датчика является минимальное такое расстояние. Слагаемое ε_t^i представляет собой погрешность измерения датчика, распределенную нормально с нулевым средним.

Определение 2.5. Облако точек с робота $C_t \subset \mathbb{R}^3$ – множество точек объектов окружающей среды в системе координат робота, полученное по показаниям датчика восприятия D_t:

$$C_t = \{D_t^i \mathbf{r_i},\ D_t^i > 0\}_{i=1}^n. \tag{2.7}$$

Определение 2.6. Одометрия представляет собой оценку перемещения робота от шага $t-1$ к шагу t, вычисленную алгоритмически по наблюдениям робота на шагах $t-1$ и t с некоторой ошибкой:

$$\widehat{\Delta S_t} = S_t \ominus S_{t-1} \oplus \Sigma_t \in SE(3), \tag{2.8}$$

где Σ_t – ошибка одометрии на шаге t, представленная в виде комбинации сдвига на нормально распределенный случайный вектор и поворота на нормально распределенный случайный угол вокруг оси Oz:

$$\Sigma_t = \left(\begin{array}{ccc|c} \cos\varepsilon_t^r & \sin\varepsilon_t^r & 0 & \varepsilon_t^x \\ -\sin\varepsilon_t^r & \cos\varepsilon_t^r & 0 & \varepsilon_t^y \\ 0 & 0 & 1 & \varepsilon_t^z \\ \hline 0 & 0 & 0 & 1 \end{array}\right); \tag{2.9}$$

$$\varepsilon_t^r \sim \mathcal{N}(0, (\sigma_t^r)^2);\ \varepsilon_t^x \sim \mathcal{N}(0, (\sigma_t^x)^2);\ \varepsilon_t^y \sim \mathcal{N}(0, (\sigma_t^y)^2);\ \varepsilon_t^z \sim \mathcal{N}(0, (\sigma_t^z)^2).$$

2.2 Метрическая модель задачи ОКЛ

Как правило, в работах, посвященных решению задачи ОКЛ, используется метрическое представление окружающей среды. В качестве такого представления может выступать облако точек, приближающее конечное подмножество точек объектов окружающей среды W_{obj}:

$$\mathcal{M}_t = \{(x_i, y_i, z_i)\}_{i=1}^{M_t} \in \mathbb{R}^{M_t \times 3}. \tag{2.10}$$

Метрическая задача картирования состоит в создании функции F, которая принимает на вход карту с предыдущего шага, облако точек и одометрию, полученные по текущему наблюдению с робота, и выдает карту на текущем шаге:

$$\begin{aligned} F: \ \mathbb{R}^{*\times 3} \times \mathbb{R}^{*\times 3} \times SE(3) \to \mathbb{R}^{*\times 3}; \\ F(\mathcal{M}_{t-1}, C_t, \widehat{\Delta S_t}) = \mathcal{M}_t; \\ \mathbb{R}^{*\times 3} = \bigcup_{n=0}^{\infty} \mathbb{R}^{n\times 3}. \end{aligned} \tag{2.11}$$

Такое облако точек может быть получено путем объединения облаков точек C_t с робота, если известны точные позы робота s_t:

$$M_t = \bigcup_{\tau=1}^{t} \bigcup_{i=1}^{n_t} S_\tau \cdot C_\tau^i. \tag{2.12}$$

В реальном мире точные позы робота неизвестны, их можно восстановить приближенно, используя данные одометрии:

$$\begin{aligned} \widehat{S}_t = S_0 \oplus \widehat{\Delta S_1} \oplus \cdots \oplus \widehat{\Delta S_t} = \\ = S_t \oplus \Sigma_1 \oplus \cdots \oplus \Sigma_t. \end{aligned} \tag{2.13}$$

Такое приближение подвержено **накоплению ошибки:** дисперсия ошибки $\Sigma_1 \oplus \cdots \oplus \Sigma_t$ увеличивается с ростом t. Пример такого накопления ошибки показан на рисунке 2.2. Для устранения накопления ошибки и коррекции карты в процессе картирования, как правило, применяются методы глобальной оптимизации.

Одним из наиболее используемых критериев качества полученного облака точек является его средняя близость к объектам окружающей среды (англ. Average Mapping Error, AME):

$$\begin{aligned} AME = \frac{1}{M_t} \sum_{i=1}^{M_t} \rho(\mathbf{m_t^i}, W_{obj}); \\ \rho(\mathbf{m}, W_{obj}) = \min_{\mathbf{w}\in W_{obj}} ||\mathbf{w} - \mathbf{m}||. \end{aligned} \tag{2.14}$$

Такой критерий может быть вычислен, например, с помощью программного пакета CloudCompare[1]. Пакет позволяет сравнить между собой два облака точек – карту и некоторое подмножество точек объектов окружающей среды. Каждой точке из первого облака ставится в соответствие ближайшая точка второго

[1]https://www.cloudcompare.org/

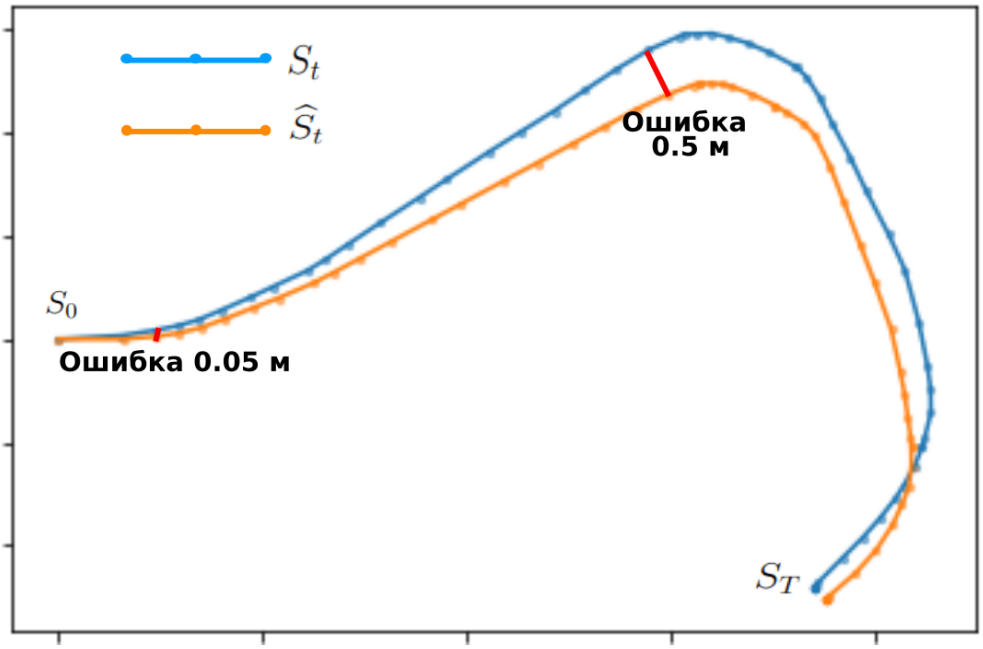

Рисунок 2.2 — Накопление ошибки позиционирования при восстановлении позы по данным одометрии.

облака. Результатом сравнения является среднее расстояние между поставленными в соответствие точками. Такой способ вычисления ошибки картирования используется в работах [42; 81]. Его основным недостатком является возможность некорректного сопоставления точек в облаках (например, точке стены в облаке, построенном алгоритмом, может быть поставлена в соответствие точка пола в истинной карте).

В работе [82] предложен метод оценки качества метрических карт, устраняющий указанный недостаток с помощью учета предсказанного локализацией ракурса камеры. В предложенном методе в соответствие точке $\mathbf{x}$ ставится та точка среды, которая была видна с ракурса $\mathbf{p_t}$ под тем же углом к главной оптической оси камеры, что и точка $\mathbf{x}$ с ракурса $\tilde{\mathbf{p}}_\mathbf{t}$. Такой способ оценки качества картирования более информативен, однако он так же не подходит для оценки качества графов локаций, поскольку элементы локаций не имеют метрических координат. В данной работе предлагается использовать комплекс показателей, определяющих пригодность построенного графа локаций для навигации.

Результатом локализации в метрической карте в момент времени t, как правило, является оценка позы робота $\widetilde{s}_t$, состоящая из оценки позиции робота $\widetilde{\mathbf{p}}_\mathbf{t}$ и оценки матрицы поворота робота $\widetilde{Q}_t$. Оценке позы $\widetilde{s}_t$ соответствует аффинное преобразование $\widetilde{S}_t$:

$$\widetilde{s}_t = (\widetilde{\mathbf{p}}_\mathbf{t}, \widetilde{Q}_t);\ \widetilde{\mathbf{p}}_\mathbf{t} = (\widetilde{p}_t^x, \widetilde{p}_t^y, \widetilde{p}_t^z) \in \mathbb{R}^3;\ \widetilde{Q}_t \in SO(3). \tag{2.15}$$

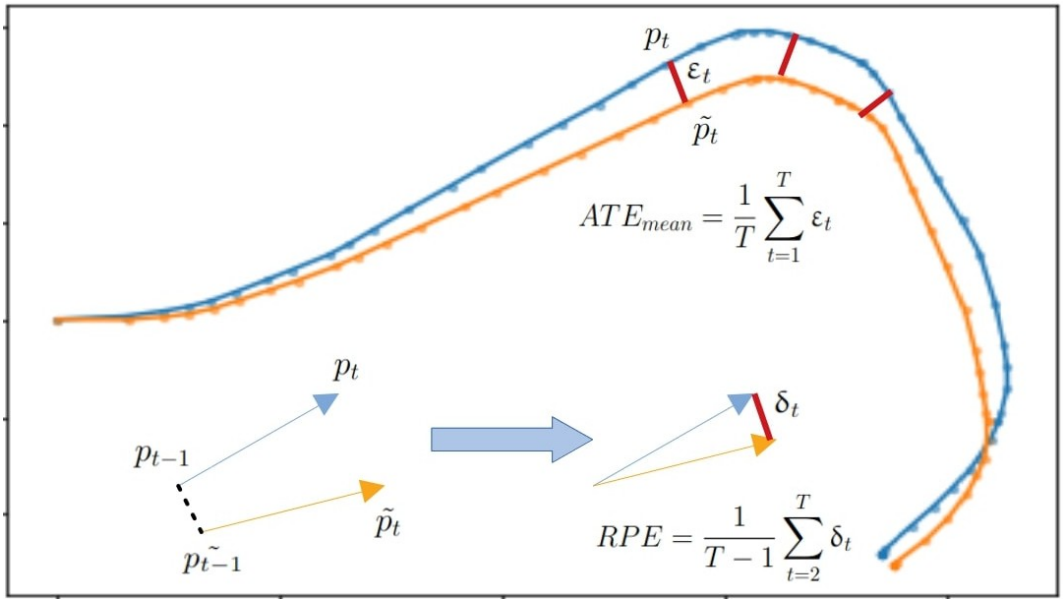

Рисунок 2.3 — Пример вычисления метрик ATE и RPE. Сниим показана истинная траектория робота, оранжевым – траектория, вычисленная методом локализации.

Качество такой метрической локализации можно оценить путем сравнения траектории, вычисленной процедурой локализации $(\tilde{p_1}, \ldots, \tilde{p_{t_{max}}})$ и истинной траекторией робота $(p_1, \ldots, p_{t_{max}})$. Как правило, для сравнения применяются два критерия качества: абсолютная ошибка траектории (англ. Absolute Trajectory Error, ATE) и относительная ошибка позиции (англ. Relative Pose Error, RPE). Вычисление значений ATE и RPE проиллюстрировано на рисунке 2.3. Критерий ATE вычисляется как усредненная ошибка между точками истинной и вычисленной процедурой локализации траектории. В качестве усредненной ошибки может выбираться среднеквадратичная (ATE_{RMSE}, от англ. Root Mean Squared Error), средняя абсолютная (ATE_{mean}), и медианная (ATE_{median}):

$$\begin{aligned} ATE_{RMSE}(\widetilde{\mathbf{p_1}}, \ldots, \widetilde{\mathbf{p_{t_{max}}}}; \mathbf{p_1}, \ldots, \mathbf{p_{t_{max}}}) &= \sqrt{\frac{1}{t_{max}}\sum_{t=1}^{t_{max}} ||\tilde{\mathbf{p}}_\mathbf{t} - \mathbf{p_t}||^2}; \\ ATE_{mean}(\widetilde{\mathbf{p_1}}, \ldots, \widetilde{\mathbf{p_{t_{max}}}}; \mathbf{p_1}, \ldots, \mathbf{p_{t_{max}}}) &= \frac{1}{t_{max}}\sum_{t=1}^{t_{max}} ||\tilde{\mathbf{p}}_\mathbf{t} - \mathbf{p_t}||; \\ ATE_{median}(\widetilde{\mathbf{p_1}}, \ldots, \widetilde{\mathbf{p_{t_{max}}}}; \mathbf{p_1}, \ldots, \mathbf{p_{t_{max}}}) &= median_{t=1}^{t_{max}} ||\tilde{\mathbf{p}}_\mathbf{t} - \mathbf{p_t}||, \end{aligned} \tag{2.16}$$

где $||\tilde{\mathbf{p}}_\mathbf{t} - \mathbf{p_t}||$ – евклидово расстояние между позициями $\tilde{\mathbf{p}}_\mathbf{t}$ и $\mathbf{p_t}$:

$$||\tilde{\mathbf{p}}_\mathbf{t} - \mathbf{p_t}|| = \sqrt{(p_t^x - \widetilde{p_t^x})^2 + (p_t^y - \widetilde{p_t^y})^2 + (p_t^z - \widetilde{p_t^z})^2}. \tag{2.17}$$

Относительная ошибка позиции RPE вычисляется как среднеквадратичная относительная ошибка между истинными перемещениями робота и перемещениями, вычисленными процедурой локализации:

$$RPE = \sqrt{\frac{1}{t_{max}-1}\sum_{t=2}^{t_{max}} \frac{||(S_t \ominus S_{t-1})_{trans} - (\widetilde{S}_t \ominus \widetilde{S_{t-1}})_{trans}||^2}{||\mathbf{p_t} - \mathbf{p_{t-1}}||^2}}, \tag{2.18}$$

где S_{trans} – вектор переноса преобразования координат S.

2.3 Топологическая модель задачи ОКЛ

В данной работе предлагается топологический подход к представлению окружающей среды – в памяти робота хранится граф локаций, каждая из которых описывает участок окружающей среды (например, комнату или лифтовый холл), охватываемый наблюдением с некоторой точки. Ребрами в графе соединяются локации, имеющие общий участок свободного пространства W_{free} (этот участок будет являться переходом между локациями). Пример разбиения пространства на локации и соединения локаций ребрами показан на рисунке 2.4.

Определение 2.7. Локацией, наблюдаемой из точки $\mathbf{p} \in W$, назовем подмножество среды W, которое включает в себя все точки объектов, находящиеся в прямой видимости из точки $\mathbf{p}$ на расстоянии не более d_{max} (дальность видимости датчиком восприятия робота), и все свободное пространство среды между точкой $\mathbf{p}$ и этими объектами:

$$\begin{aligned} d_{obj}(\mathbf{p}, \mathbf{r}) &= \min\{\delta : \ \mathbf{p} + \delta\mathbf{r} \in W_{obj}\}; \\ ray(\mathbf{p}, \mathbf{r}, d_{max}) &= \{\mathbf{p} + \alpha\mathbf{r} : \ 0 \leqslant \alpha \leqslant d_{max}, \alpha \leqslant d_{obj}(\mathbf{p}, \mathbf{r})\}; \\ loc = Location(\mathbf{p}, d_{max}) &= \bigcup_{\mathbf{r} \in B_1(0)} ray(\mathbf{p}, \mathbf{r}, d_{max}) \subset W; \\ loc_{obs} &= \mathbf{p} \in loc. \end{aligned} \tag{2.19}$$

Точка p называется **точкой наблюдения** локации.

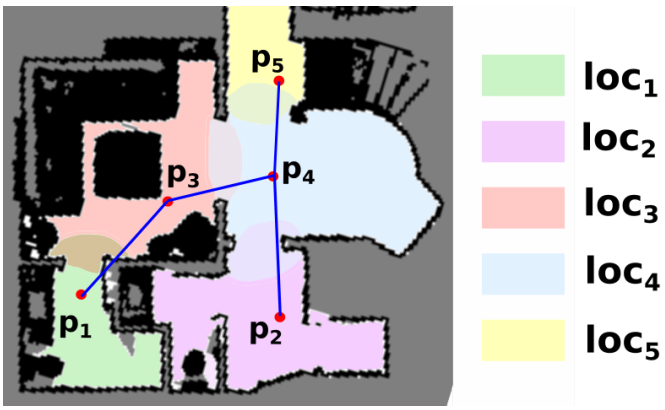

Рисунок 2.4 — Пример разбиения пространства на локации (красным показаны точки наблюдения локаций, смежные локации соединены синими ребрами).

Определение 2.8. Две локации loc и loc' называются **смежными**, если их пересечение содержит точки свободного пространства среды:

$$adjacency(loc, loc') \iff loc \cap loc' \cap Wfree \neq \varnothing. \tag{2.20}$$

Задача топологического картирования заключается в построении графа локаций $G_t = (V_t, E_t)$ по облакам точек и данным одометрии с робота в реальном времени. Алгоритм решения задачи на каждом шаге t принимает облако точек и одометрию, полученные с робота, а также граф локаций с предыдущего шага $G_{t-1} = (V_{t-1}, E_{t-1})$, и выдает обновленный граф локаций $G_t = (V_t, E_t)$:

$$\begin{aligned} A(C_t, \widehat{\Delta S_t}, G_{t-1}) = G_t = (V_t, E_t); \\ V_t = (loc_1 \subset W, \ldots, loc_n \subset W); \\ E_t = (e_1 \in V_t \times V_t \times \mathbb{R}^+, \ldots, e_m \in V_t \times V_t \times \mathbb{R}^+), \end{aligned} \tag{2.21}$$

где V_t – локации, добавленные в граф, а E_t – ребра графа. Каждое ребро $e = (u \in V_t, v \in V_t, l)$ помимо локаций u и v, которые оно соединяет, содержит оценку расстояния l между точками наблюдения локаций u_{obs} и v_{obs}. Каждая локация в графе $loc \in V_t$ наблюдается с некоторой пройденной роботом точки:

$$\forall loc \in V_t \; \exists \boldsymbol{\tau} \leqslant t : \; loc = Location(\mathbf{p}_\tau, obs_\tau). \tag{2.22}$$

Для успешной и эффективной навигации с использованием построенного графа локаций необходимы успешная локализация робота в графе, успешное построение пути от точки старта робота до целевой точки и движение вдоль этого пути. В данной работе качество топологического картирования предлагается оценивать с точки зрения **путевой эффективности**, которая зависит от оптимальности и допустимости путей, построенных по графу.

Определение 2.9. Путь по графу $G = (V, E)$ между точками среды $\mathbf{p_s} \in W, \mathbf{p_f} \in W$ определим следующим образом: найдем локации loc_s, loc_f, такие, что $\mathbf{p_s} \in loc_s$, $\mathbf{p_f} \in loc_f$, имеющие ближайшие точки наблюдения к точкам $\mathbf{p_s}$ и $\mathbf{p_f}$ соответственно. Построим кратчайший путь между локациями loc_s и loc_f по рёбрам графа E, используя расстояния, записанные на ребрах. Если локации loc_s и loc_f совпадают, путь по графу G состоит только из точек $\mathbf{p_s}$ и $\mathbf{p_f}$. Если не совпадают – путь по графу G включает в себя точки наблюдения всех локаций найденного кратчайшего пути между точками $\mathbf{p_s}$ и $\mathbf{p_f}$.

$$
\begin{gathered}
loc_s = \arg \min_{loc \in V;\ \mathbf{p_s} \in loc} ||\mathbf{loc_{obs}} - \mathbf{p_s}||; \\
loc_f = \arg \min_{loc \in V;\ \mathbf{p_f} \in loc} ||\mathbf{loc_{obs}} - \mathbf{p_f}||; \\
w(u, v) = \min_{(u,v,l) \in E} l; \\
\pi = \arg \min_{\substack{\pi = (\pi_0, \ldots, \pi_k) \subset V; \\ \pi_0 = loc_s, \pi_k = loc_f; \\ \forall i = 1, \ldots, k \hookrightarrow (\pi_{i-1}, \pi_i) \in E}} \sum_{i=1}^{k} w(\pi_{i-1}, \pi_i); \\
Path(\mathbf{p_s}, \mathbf{p_f}, G_t) = \begin{cases} (\mathbf{p_s}, \mathbf{p_f}) & loc_s = loc_f \\ (\mathbf{p_s}, (\pi_0)_{obs}, \ldots, (\pi_k)_{obs}, \mathbf{p_f}) & loc_s \neq loc_f \end{cases}.
\end{gathered}
\tag{2.23}
$$

Определение 2.10 Путь по графу π назовем **допустимым**, если каждое его звено соединяет смежные локации. Множество всех допустимых путей обозначим как P^+:

$$
\begin{gathered}
P^+ = \{\pi = (\pi_0, \ldots, \pi_k) \in 2^W : \\
Location(\pi_{i-1}) \cap Location(\pi_i) \cap W_{free} \neq \varnothing \ \forall i = 1, \ldots, k\}.
\end{gathered}
\tag{2.24}
$$

Определение 2.11 **Длиной** пути по графу $\pi = (\pi_0, \ldots, \pi_k)$ назовем сумму длин кратчайших путей в среде, соединяющих его промежуточные точки:

$$
\begin{gathered}
len_{W_{free}}(\mathbf{s} \in W_{free}, \mathbf{g} \in W_{free}) = \\
= \inf_{\pi = (\pi_0 = s, \ldots, \pi_k = g) \subset W_{free};\ \forall i = 1, \ldots, k \hookrightarrow [\pi_{i-1}, \pi_i] \subset W_{free}} \sum_{i=1}^{k} ||\pi_i - \pi_{i-1}||; \\
len(\pi) = \sum_{i=1}^{k} len_{W_{free}}(\pi_{i-1}, \pi_i).
\end{gathered}
\tag{2.25}
$$

Определение 2.12 Определенное выше значение $len_{W_{free}}$ назовем **расстоянием по среде** W_{free} между точками s и g. Пример построения пути по графу и расчета его длины показан на рисунке 2.5.

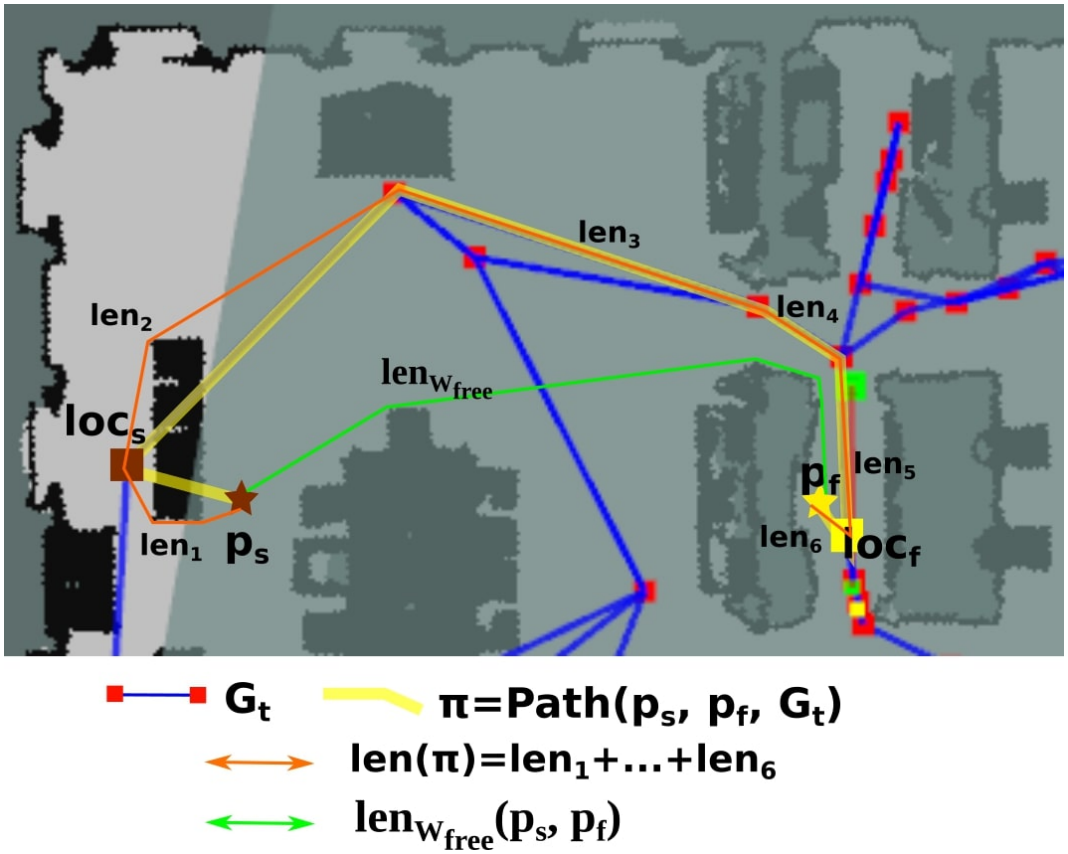

Рисунок 2.5 — Пример построения пути по графу между двумя точками среды и вычисления его длины. Препятствия среды показаны черным, свободные области – светло-серым.

Определение 2.13 **Эффективностью** допустимого пути по графу $\pi = (x_0, \ldots, x_n)$ назовем отношение расстояния по среде между точками x_0 и x_n к длине пути π. Эффективность недопустимого или ненайденного пути положим равной нулю. Обозначим значение эффективности пути π как $SPL(\pi)$ (от англ. Success weighted by Path Length):

$$SPL(\pi) = \frac{len_{W_{free}}(x_0, x_n)}{len(\pi)} \cdot I(\pi \in P^+). \tag{2.26}$$

Здесь и далее $I(condition)$ обозначает оператор, который возвращает 1, если условие $condition$ выполняется, и возвращает 0 в противном случае. Целевым критерием качества является средняя эффективность путей по графу между

всеми парами точек на траектории робота:

$$SPL(W, G_{t_{max}}) = \frac{1}{(t_{max}+1)^2} \sum_{t_1=0}^{t_{max}} \sum_{t_2=0}^{t_{max}} SPL(path(\mathbf{p_{t_1}}, \mathbf{p_{t_2}}, G_{t_{max}})) \tag{2.27}$$

Так как значение t_{max} может быть велико (порядка нескольких тысяч), вычисление значения $SPL(W, G)$ может занять много времени и вычислительных ресурсов (например, в случае $t_{max} = 1000$ потребуется более 1 млн вычислений пути по графу и расстояния по среде, что может занять до нескольких недель вычислений на стандартном компьютере). Для ускорения расчетов путевую эффективность можно вычислить приблизительно, выбрав случайным образом N пар точек ($s_i \in W_{free}, g_i \in W_{free}$). Обозначим вычисленный таким образом результат как SPL_N:

$$SPL_N(W, G) = \frac{1}{N} \sum_{i=1}^{N} SPL(path(s_i, g_i, G)). \tag{2.28}$$

Вычисление значения SPL_N для больших N все еще достаточно затратно по вычислениям (например, для $N = 1000$ вычисления на стандартном компьютере займут от нескольких минут до нескольких часов), поэтому для оценки качества картирования в данной работе рассматривается ряд других показателей, влияющих напрямую на эффективность путей:

1. Количество компонент связности N_{comp}. Между точками, оказавшимися в локациях, относящихся к разным компонентам связности, путь не будет найден, что сделает успешную навигацию между ними невозможной.
2. Доля покрытия свободного пространства среды локациями графа. До точек, не принадлежащих ни к одной локации, путь не будет найден, что сделает невозможной успешную навигацию до них. Для оценки навигационных возможностей покрытие вычисляется по всем локациям, входящим в наибольшую компоненту связности графа:

$$Coverage = \frac{|\bigcup_{loc \in V_{main}} loc|}{|\bigcup_{t=0}^{t_{max}} Location(\mathbf{p_t})|}, \tag{2.29}$$

где V_{main} обозначает наибольшую компоненту связности графа локаций, а $|W|$ обозначает объем области пространства W.

3. Корректность ребер, добавленных в граф. Прокладка пути через ребро, соединяющее несмежные локации, приведет к сбою навигации при попытке перейти по этому ребру. Корректность оценивается как доля ребер, соединяющих смежные локации, среди всех ребер графа:

$$Correctness = \frac{\sum\limits_{(loc,loc')\in E_{t_{max}}} I(loc \cap loc' \cap W_{free} \neq \varnothing)}{|E_{t_{max}}|}. \tag{2.30}$$

4. Полнота графа. Отсутствие ребра между смежными локациями loc и loc' снижает эффективность путей между точками, находящимися в этих локациях. Полнота графа вычисляется как доля ребер, соединяющих смежные локации, среди всех пар смежных локаций:

$$Recall = \frac{\sum\limits_{(loc,loc')\in E_{t_{max}}} I(loc \cap loc' \cap W_{free} \neq \varnothing)}{|(loc \in V, loc' \in V) : loc \cap loc' \cap W_{free} = \varnothing|}. \tag{2.31}$$

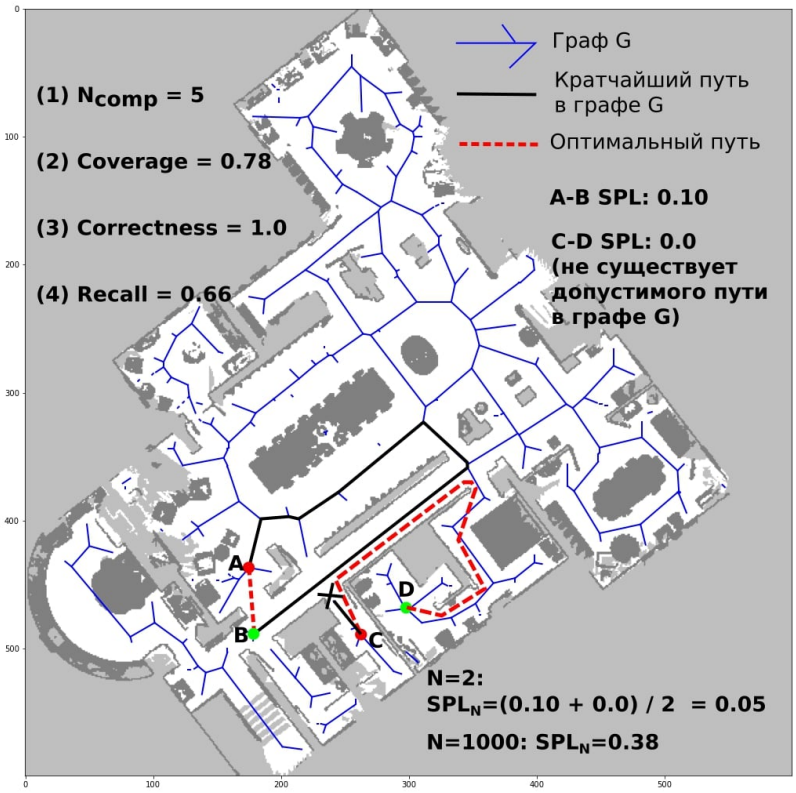

Рисунок 2.6 — Пример вычисления значения SPL_N по графу локаций по двум парам точек (A-B, C-D).

Пример вычисления значения критериев N_{comp}, $Coverage$, $Correctness$, $Recall$, а также значения SPL_N для $N = 2$ представлен на рисунке 2.6.

Задача локализации

Задача локализации в топологической карте, представляемой в виде графа $G_t = (V_t, E_t)$, состоит в следующем: на каждом шаге t по облаку точек и одометрии с робота определить локацию $v^t_{cur} \in V_t$, в которой находится робот в данный момент, и определить положение робота относительно точки наблюдения этой локации $T^t_{cur} \in SE(3)$.

$$\begin{gathered} Alg(C_t, \widehat{\Delta S_t}, G_t) = (v^t_{cur}, T^t_{cur}); \\ v^t_{cur} \in V_t;\ T^t_{cur} \in SE(3). \end{gathered} \tag{2.32}$$

В данной работе задача ОКЛ рассматривается в топологической постановке, в которой метрические координаты в явном виде не используются. Для вычисления значений ATE по результатам локализации, представленным в виде локации и положения относительно ее точки наблюдения (v^t_{cur}, T^t_{cur}), предлагается вычислять метрический результат локализации с использованием информации о положении точки наблюдения локации v_{cur}, и далее для метрической оценки качества локализации вычислять среднюю абсолютную ошибку ATE_{mean}:

$$\begin{gathered} S_p = \left(\begin{array}{ccc|c} 1 & 0 & 0 & p^x \\ 0 & 1 & 0 & p^y \\ 0 & 0 & 1 & p^z \\ \hline 0 & 0 & 0 & 1 \end{array}\right); \\ \widetilde{S}_t = S_{(v^t_{cur})_{obs}} \oplus T^t_{cur}; \\ ATE = ATE_{mean}(\widetilde{S}_1 \cdot \mathbf{0}, \ldots, \widetilde{S_T} \cdot \mathbf{0}; \mathbf{p_1}, \ldots, \mathbf{p_t}). \end{gathered} \tag{2.33}$$

Во многих алгоритмах локализации в топологической карте поиск локации v_{cur} и положения относительно ее точки наблюдения T_{cur} выполняется последовательно, и при неверном определении локации v_{cur} метрическая ошибка локализации оказывается сравнимой с размером окружающей среды W. Таким образом, ключевым показателем является точность определения локации v_{cur}. Обозначим этот показатель как успешность локализации SR_{loc} (от англ. Success Rate):

$$SR_{loc} = \frac{\sum\limits_{t=1}^{t_{max}} I(\mathbf{p_t} \in v^t_{cur})}{t_{max}}, \tag{2.34}$$

Пример вычисления значения SR_{loc} в сравнении с метрической ошибкой ATE_{mean} показан на рисунке 2.7.

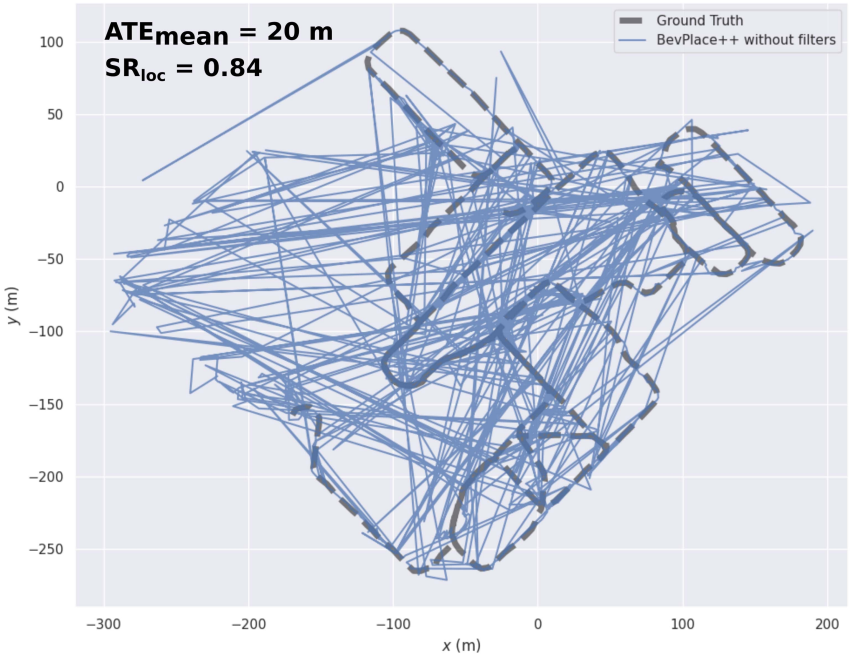

Рисунок 2.7 — Пример вычисления значений критериев качества локализации ATE_{mean} и SR_{loc}. Истинная траектория робота показана черной пунктирной линией; траектория, вычисленная алгоритмом локализации BEVPlace++ [77] – синей сплошной линией.

2.4 Выводы по главе

В данной главе для решения проблемы оценки качества топологических методов ОКЛ предложена математическая модель задачи топологического картирования и локализации. Входными данными в предложенной модели являются облако точек и одометрия, полученные по наблюдениям с робота. Результатом топологического картирования является граф локаций (участков среды, охватываемых одним наблюдением). Результатом локализации является состояние робота в графе – текущая локация и положение в ней.

В главе определена методика построения пути по графу локаций между двумя точками окружающей среды и даны определения допустимости и эффективности пути по графу. Предложен критерий качества графа локаций, вычисляемый как средняя эффективность всех построенных по нему путей и обозначаемый как SPL. Значение предложенного критерия зависит как от оптимальности, так и от допустимости путей. Предложенный способ оценки качества карты может быть применен как для метрических методов ОКЛ,

использующих графовые структуры (например, граф позиций), так и для топологических методов ОКЛ.

Помимо путевой эффективности, в главе предлагается ряд других, легко вычислимых критериев:

1. Количество компонент связности графа локаций;
2. Доля покрытия среды локациями, входящими в наибольшую компоненту связности;
3. Корректность ребер (доля ребер графа, соединяющих смежные локации);
4. Полнота ребер (отношение количества ребер графа, соединяющих смежные локации, к общему количеству пар смежных локаций).

Вычисление значения SPL_N и других предложенных критериев позволяет оценить, насколько построенный граф локаций пригоден для планирования маршрутов и навигации робота.

В главе предложен критерий оценки качества локализации в графе локаций – средняя успешность локализации. Она вычисляется как доля локаций, выданных алгоритмом ОКЛ в качестве результата локализации, которые действительно включают в себя текущую позицию робота. Такой критерий позволяет учитывать специфику задачи топологического ОКЛ и оценивать качество локализации даже в том случае, если точное метрическое положение робота внутри локации алгоритмом не вычисляется.

Глава 3. Разработка алгоритма топологического картирования и локализации

В рамках данной работы для решения поставленной в разделе 2.3 задачи был разработан оригинальный алгоритм топологического картирования и локализации, названный PRISM-TopoMap. Алгоритм строит топологическую карту в виде графа локаций и на каждом шаге определяет состояние робота в графе – локацию, в которой он находится, и его положение внутри локации. В работе был предложен двухуровневый подход к планированию пути до целевой точки с использованием построенной топологической карты. В главе приводится подробное описание составных частей алгоритма PRISM-TopoMap, а также описание предложенного двухуровневого подхода к планированию пути.

3.1 Общая схема алгоритма

Алгоритм PRISM-TopoMap строит топологическую карту в виде графа локаций по облакам точек с лидара робота (дополнительно к облаку точек могут быть добавлены изображения с передней и задней камер робота) и одометрии с любого источника. Алгоритм состоит из двух выполняемых параллельно процедур: процедура построения и поддержания графа локаций и процедура локализации. Его общая схема показана на рисунке 3.1.

Алгоритм решает задачу, поставленную в разделе 2.3. На вход алгоритму на каждом шаге t подается лидарное облако точек C_t и оценка перемещения робота $\widehat{\Delta S_t}$ (одометрия), а также граф локаций G_{t-1}. Выходом алгоритма является обновленный граф локаций $G_t = (V_t, E_t)$ и текущее состояние робота в графе: v^t_{cur} – вершина, в которой находится робот в момент t, и T^t_{cur} – положение робота относительно точки наблюдения локации v_{cur}:

$$\begin{gathered} Alg(C_t, \widehat{\Delta S_t}, G_{t-1}) = (G_t, v^t_{cur}, T^t_{cur}); \\ v^t_{cur} \in V_t; T^t_{cur} \in SE(3). \end{gathered} \tag{3.1}$$

Для каждой локации предложенный алгоритм сохраняет дескриптор, предсказанный нейросетью для распознавания места, а также двумерную про-

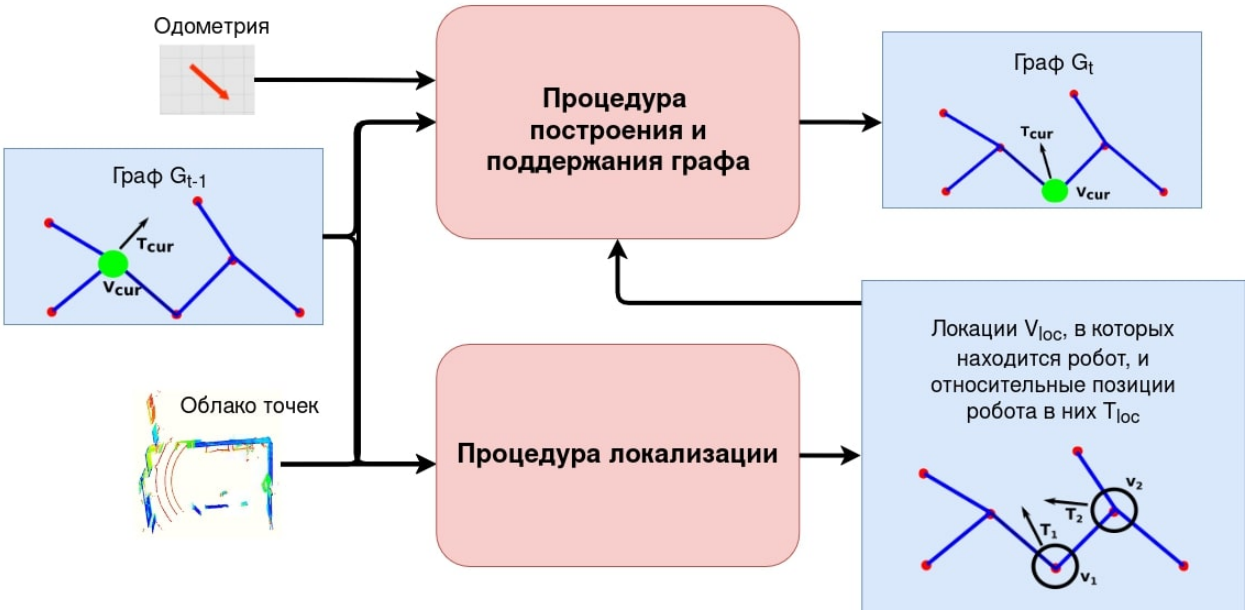

Рисунок 3.1 — Общая схема разработанного алгоритма топологического картирования и локализации.

екцию облака точек для поиска относительного положения T^t_{cur}:

$$F(loc) = (desc_{loc}; scan_{loc} = projection(C_{loc})). \tag{3.2}$$

Для каждого ребра графа сохраняются приблизительные положения конца ребра относительно его начала. Эта информация используется для построения путей по графу:

$$F(e = (u,v)) \approx v_{obs} - u_{obs}. \tag{3.3}$$

Для навигации в графе локаций, построенном с помощью разработанного алгоритма, предлагается использовать двухуровневое планирование пути. На верхнем уровне в графе локаций строится глобальный путь от текущей локации v_{cur} до локации, содержащей целевую точку. Для поиска такого пути применяется алгоритм Дейкстры [83], в качестве весов ребер используются их длины, полученные по приписанным к ним относительным позициям. На нижнем уровне строится локальный путь с учетом препятствий до точки наблюдения следующей локации в глобальном пути (либо, если робот находится в одной локации с целевой точкой – до целевой точки). Схема предложенного подхода к планированию пути изображена на рисунке 3.2. Движение вдоль построенного пути с избеганием статических и динамических препятствий может осуществляться, например, с помощью методов модельно-предиктивного управления [84; 7], в которых построенный путь будет использоваться в качестве начального приближения.

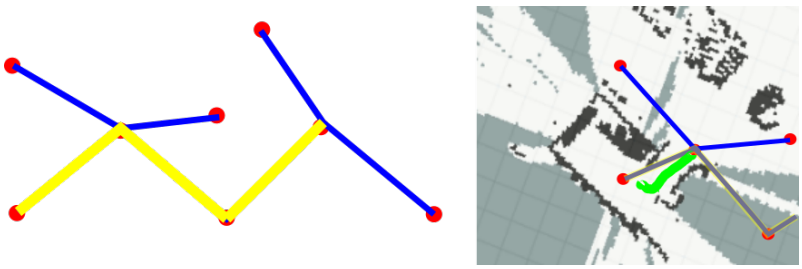

Рисунок 3.2 — Схема предложенного двухуровневый подход к планированию пути: глобальный топологический путь на верхнем уровне (слева) и локальный метрический путь на нижнем уровне (справа, показан зеленым).

Граф и текущее состояние робота в нем обновляются в зависимости от одометрии, перекрытия текущего облака точек с робота с локацией v_{cur} и результатов локализации. Процедура обновления графа и состояния (v_{cur}, T_{cur}) и процедура локализации в графе подробно описаны ниже.

3.2 Процедура локализации в топологической карте

Процедура локализации в графе $G_t = (V_t, E_t)$ принимает на вход текущее наблюдение с робота C_t (облако точек, к которому дополнительно могут быть добавлены изображения с камер робота), а также граф локаций с предыдущего шага G_{t-1} и выдает список локаций в графе, в которых может находиться робот в момент t, с оценкой положения робота относительно их точек наблюдения. Псевдокод процедуры локализации представлен в алгоритме 1.

Локализация проводится в два этапа: сначала с помощью методов распознавания места находится список локаций-кандидатов, чьи сканы близки к текущему скану робота в пространстве дескрипторов. Далее с помощью оригинального алгоритма сопоставления сканов отбрасываются ложноположительные результаты распознавания места (т.е. близкие к текущему наблюдению с робота по дескрипторам, но не содержащие позицию робота $\mathbf{p_t}$ локации), а для всех оставшихся локаций вычисляется положение робота относительно их точек наблюдения. Общая схема процедуры локализации представлена на рисунке 3.3. Ниже приведено подробное описание двух этапов локализации.

Алгоритм 1: Процедура локализации

Вход: Облако точек C_t; граф локаций $G_{t-1} = (V_{t-1}; E_{t-1})$
Выход: Список локаций в графе, в которых может находиться робот, с оценками положения робота относительно их точек наблюдения:
$V^t_{loc} = \{v^{t,i}_{loc}, i = 1, \ldots, k\}; \mathcal{T}^t_{loc} = \{T^{t,i}_{loc}, i = 1, \ldots, k\}$.
Параметры: количество локаций k; порог успешного сопоставления th
$desc_t = F_{PR}(C_t)$/* Вычисляем дескриптор текущего наблюдения с робота */
D = $\{desc_{loc} : loc \in V_{t-1}\}$/* Дескрипторы всех локаций графа */
V^t_{found} = GetKNearestTo(V_{t-1}, D, $desc_t$, k)/* Выбираем k локаций из графа, чьи дескрипторы наиболее близки к декскриптору $desc_t$ */
$scan_t$ = Projection(C_t)/* Проекция текущего облака точек с робота */
$V^t_{loc} = \varnothing; \mathcal{T}^t_{loc} = \varnothing$
for $v \in V^t_{found}$ **do**
  T_{loc}, matchScore = ScanMatching($scan_v$, $scan_t$)/* Сопоставляем проекцию текущего облака точек с робота с проекцией локации v */
  if $matchScore > th$ **then**
    $V^t_{loc} = V^t_{loc} \cup \{v\}$
    $\mathcal{T}^t_{loc} = \mathcal{T}^t_{loc} \cup \{T_{loc}\}$

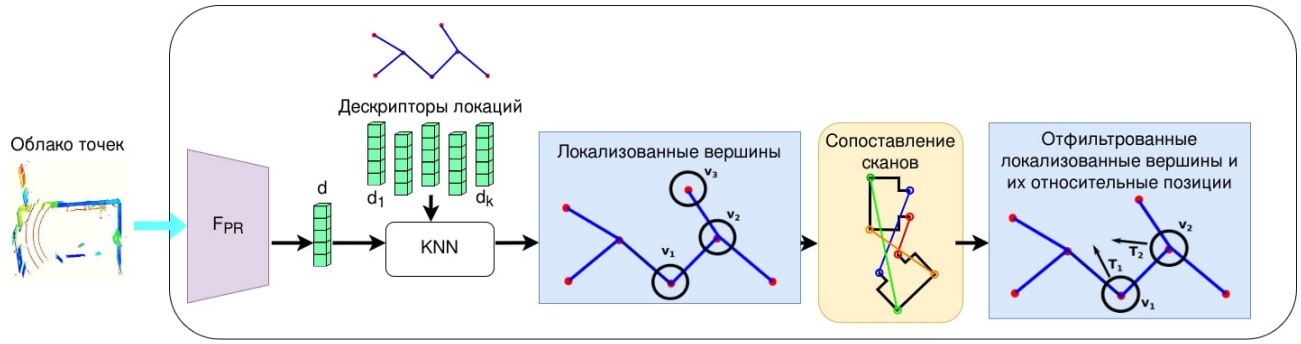

Рисунок 3.3 — Схема процедуры локализации.

Глобальный поиск локации Для поиска локаций, содержащих текущее положение робота, используются методы распознавания места. По текущему наблюдению с робота с помощью нейронной сети вычисляется дескриптор – вектор признаков небольшой (по сравнению с размерностью облака точек) размерности:

$$F_{PR}(C_t) = desc_t \in \mathbb{R}^N, \tag{3.4}$$

где F_{PR} – нейросетевая модель для вычисления дескриптора облака точек. Дескрипторы предварительно вычисляются для каждой локации при добавлении локации в граф. Далее в графе вычисляются k локаций, чьи дескрипторы наиболее близки к дескриптору текущего наблюдения по евклидову расстоянию.

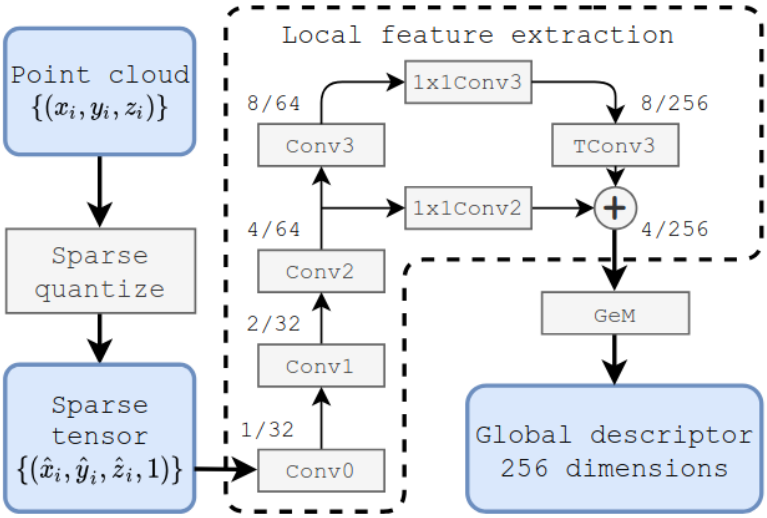

Рисунок 3.4 — Схема нейросетевой модели MinkLoc3D. Источник [65].

В случае, когда на роботе в качестве наблюдения доступно только облако точек с лидара, в качестве нейросети F_{PR} применяется модель MinkLoc3D [65]. Она представляет собой сверточную нейронную сеть, принимающую на вход предварительно дискретизованное облако точек. Дискретизация заключается в округлении координат точек облака до узлов сетки с фиксированным шагом (значение каждой координаты делится на шаг сетки, затем округляется по правилам математики до целого, затем умножается на шаг сетки) и удалении точек с одинаковыми координатами. Выходом нейросети является вектор размерности 256, представляющий собой дескриптор облака точек. Схема нейросети MinkLoc3D представлена на рисунке 3.4.

Если на роботе помимо облака точек с лидара доступны изображения с передней и задней камер, то в качестве нейросети F_{PR} применяется модель MSSPlace [72]. MSSPlace представляет собой мультимодальное расширение модели MinkLoc3D. На вход модели подается облако точек и пара изображений с робота. Модель состоит из двух последовательностей сверточных слоев – вычисление дескриптора по облаку точек и по изображениям соответственно. Вычисление дескриптора по облаку точек производится нейронной сетью с архитектурой, взятой из модели MinkLoc3D. Вычисление дескриптора по облаку точек производится сверточной нейронной сетью, состоящей из кодировщика изображений на основе сети ResNet-18, а также слоев усреднения и понижения дискретизации GeM Pooling [85]. Итоговый дескриптор наблюдения получается путем конкатенации дескриптора облака точек и дескриптора пары

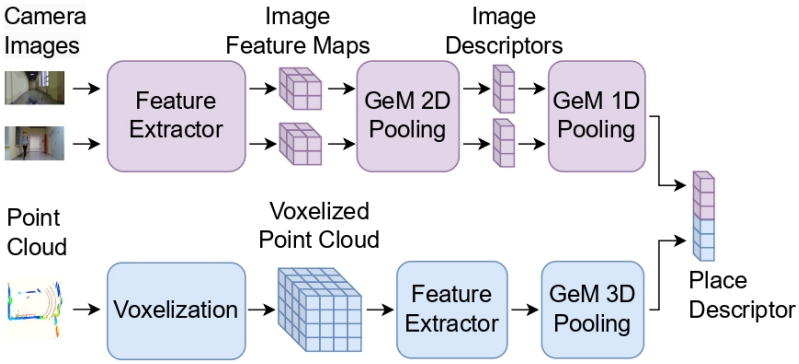

Рисунок 3.5 — Схема нейросетевой модели MinkLoc3D. Источник [72].

изображений и имеет размерность 512. Схема архитектуры нейросети MSSPlace представлена на рисунке 3.5.

Результатом первого этапа локализации является список локаций-кандидатов, состоящий из k локаций, чьи дескрипторы наиболее близки по евклидовой метрике к дескриптору текущего наблюдения:

$$\begin{aligned} dist_{loc} &= ||desc_{loc} - desc_t||_2; \\ V^t_{found} &= \{v \in V_{t-1} : |\{dist_{loc} : loc \in V_{t-1}\} \cap [0, dist_v]| < k\}. \end{aligned} \tag{3.5}$$

Фильтрация результатов глобального поиска и вычисление относительной позиции Из-за внешней схожести облаков точек и специфики обучающего набора данных для нейросети F_{PR}, окружающей среды и датчиков робота, методы глобальной локализации, основанные на нейросетевом распознавании места, могут выдавать ложноположительные результаты:

$$loc \in V^t_{found};\ IoU(C_t, C_{loc}) = 0. \tag{3.6}$$

В таком случае положение робототехнической системы в пространстве может определяться некорректно, что может привести к добавлению в граф ребер между удаленными друг от друга локациями и, как следствие, к сбоям навигации. Для устранения таких ситуаций в рамках диссертационной работы был разработан оригинальный алгоритм сопоставления сканов, который выявляет и отсеивает ложноположительные результаты глобальной локализации и вычисляет относительную позицию для каждой корректно найденной локации.

Для сопоставления используются двумерные проекции лидарных облаков точек на горизонтальную плоскость. Двумерная проекция берется по всем точкам облака, удаленным на расстояние не более r от точки наблюдения и представляется в виде целочисленной матрицы, в которой каждая ячейка имеет

значение 0, 1 или 2 и содержит информацию о точках облака в соответствующем столбике размера $d \times d$:

$$\begin{aligned} scan_t = proj_{d,r}(C_t) \in \{0,1,2\}^{\frac{2r}{d} \times \frac{2r}{d}}; \\ col_{i,j} = \{(x,y,z) \in C_t; \\ x \in [(i - \frac{r}{d}) \cdot d; (i + 1 - \frac{r}{d}) \cdot d]; \\ y \in [(j - \frac{r}{d}) \cdot d; (j + 1 - \frac{r}{d}) \cdot d]\} \end{aligned} \tag{3.7}$$

$$z_{max}(i,j) = \max_{(x,y,z) \in col_{i,j}} z;$$

$$scan_{t;i,j} = \begin{cases} 0, col_{i,j} = \varnothing \\ 1, z_{max}(i,j) \in (-\infty, z_{floor}) \cup (\ z_{ceil}, \infty) \\ 2, z_{floor} <= z_{max}(i,j) <= z_{ceil}, \end{cases} \tag{3.8}$$

где z_{floor} – максимальная высота точек поверхности, z_{ceil} – минимальная высота точек потолка.

Так достигается значительная экономия памяти, потребляемой картой, поскольку с использованием двумерных проекций для сопоставления исчезает необходимость хранить в карте исходные облака точек. Облако точек в оперативной и постоянной памяти занимает от 1 до 15 МБ, в зависимости от разрешения лидара, в то время как его двумерная проекция размером 360x360 занимает 130 кБ в оперативной памяти и порядка 10 кБ в постоянной (при сохранении в виде изображения в формате .png).

Для сопоставления по обоим облакам точек берется двумерная проекция в соответствии с формулой 3.8 и преобразуется в формат черно-белого изображения. Далее из этих изображений извлекаются особые точки и их дескрипторы алгоритмом ORB [86]. Извлеченные особые точки сопоставляются по дескрипторам с помощью алгоритма FLANN [87]. Затем проводится итеративная процедура удаления выбросов среди сопоставленных дескрипторов и вычисления итогового преобразования координат согласно алгоритму 2. Пример сопоставления облаков точек с помощью представленного алгоритма показан на рисунке 3.6.

На каждой итерации по текущему набору сопоставленных пар особых точек методом наименьших квадратов вычисляется преобразование координат, сопоставляющее сканы. Далее из набора удаляются те пары особых точек, на

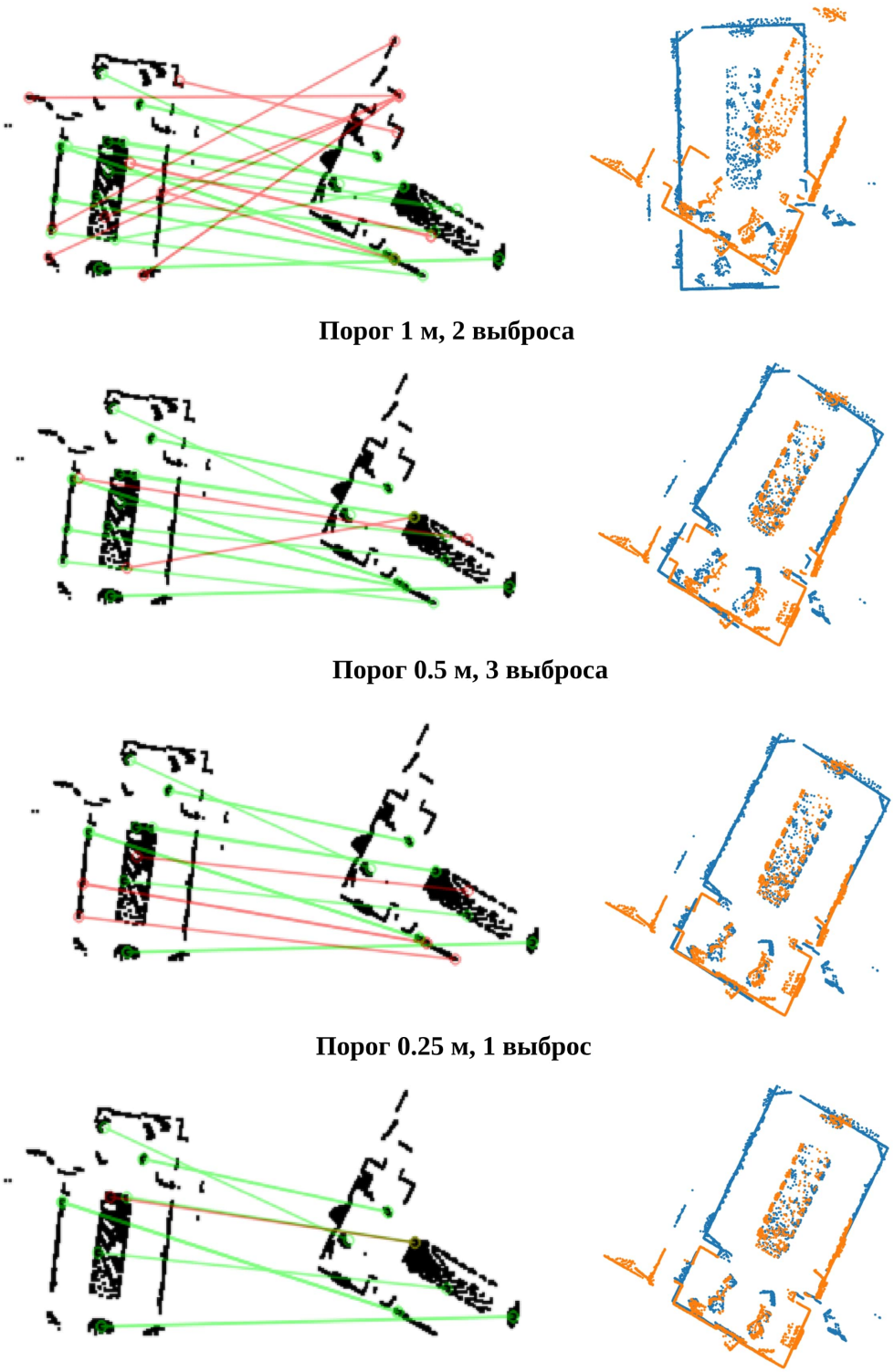

Рисунок 3.6 — Пример сопоставления проекций облаков точек с помощью разработанного алгоритма. Слева изображены пары сопоставленных особых точек, справа – применение преобразования координат, полученного после каждой итерации. Зеленым показаны оставшиеся после текущей итерации пары, красным – удаляемые на текущей итерации.

Алгоритм 2: Вычисление преобразования координат, сопоставляющего облака точек, и удаление выбросов среди сопоставленных особых точек.

Вход: Координаты попарно сопоставленных особых точек на изображениях $M = \{(p_i \in \mathbb{R}^2, q_i \in \mathbb{R}^2)\}_{i=1}^{N}$
Выход: Преобразование координат T_{loc} между облаками точек
Параметры: Максимальное число итераций max_iter; пороги сопоставления $\delta_{iter}, iter = 1, \ldots, max\_iter$; минимальное число сопоставлений K
for $iter = 1...max\_iter$ **do**
    if $|M| < K$ **then**
        return NULL
    $T = LeastSquareTransform(M)$
    for $i = 1...N$ **do**
        if $||Tp_i - q_i|| > \delta_{iter}$ **then**
            $M.remove((p_i, q_i))$
$T_{loc} = LeastSquareTransform(M)$

которых расхождение после применения преобразования превышает порог. Значения порогов на итерациях задаются невозрастающей последовательностью, таким образом, с каждой итерацией повышается точность найденного сопоставления в случае его нахождения.

Для оценки корректности найденного преобразования координат T_{loc} вычисляется оценка качества сопоставления $matchScore$. Она вычисляется следующим образом: преобразование T_{loc} применяется к первому облаку точек, затем по обоим облакам вычисляется дискретизованная проекция по формуле 3.8. Итоговая оценка рассчитывается как отношение количества совпавших ячеек препятствий к количеству несовпадений (ячеек, в которые на одном из облаков попали точки препятствий, а на втором – точки пола или потолка). Оценка учитывает долю перекрытия двух облаков – итоговое значение $matchScore$ умножается на корень четвертой степени из значения IoU:

$$matchScore = \frac{goodMatch}{goodMatch + badMatch} \cdot IoU^{1/4}. \tag{3.9}$$

Процедура вычисления $matchScore$ представлена в алгоритме 3.

Алгоритм 3: Вычисление оценки качества сопоставления двух облаков точек.

Вход: Облака точек C_t и C_v; преобразование координат T_{loc} от облака точек C_t к облаку C_v
Выход: $matchScore$ – оценка качества сопоставления
$C_t = T_{loc} \cdot C_t$ /* Применение преобразования T_{loc} к облаку точек C_t */
$scan_t = proj_{d,r}(C_t)$
$scan_v = proj_{d,r}(C_v)$
$goodMatch = 0; badMatch = 0$
$intersection = 0; union = 0$
for $i = 1, \ldots, \frac{2r}{d}$ **do**
    if $scan_{t;i,j} == scan_{v;i,j} == 2$ **then**
        $goodMatch = goodMatch + 1$
    if $scan_{t;i,j} \cdot scan_{t;i,j} == 2$ **then**
        $badMatch = badMatch + 1$
    if $scan_{t;i,j} \cdot scan_{v;i,j} > 0$ **then**
        $intersection = intersection + 1$
    if $scan_{t;i,j} > 0$ ***or*** $scan_{v;i,j} > 0$ **then**
        $union = union + 1$
$IoU = intersection/union$
$matchScore = \frac{goodMatch}{goodMatch+badMatch} \cdot IoU^{1/4}$

3.3 Процедура построения и обновления топологической карты

Процедура поддержания графа строит и обновляет топологическую карту (граф локаций) исходя из наблюдений с бортовых датчиков робота, данных одометрии и результатов модуля локализации. Каждой локации $loc \in V_t$ приписывается двумерная проекция ее облака точек, а также ее дескриптор, извлеченный нейросетевой моделью распознавнаия места по наблюдению локации loc_{obs}. Смежные локации соединяются ребрами, и ребрам приписываются относительные позиции между точками наблюдения локаций, которые они соединяют.

Результатом работы процедуры на шаге t является граф локаций G_t, покрывающий все пространство, посещенное роботом, и текущее состояние робота в графе: локация v^t_{cur}, в которой находится робот на шаге t, и положение робота относительно ее точки наблюдения T^t_{cur}. На вход процедуре подаются граф

локаций G_{t-1}, состояние на шаге $t-1$, результаты локализации, измерение одометрии (оценка перемещения робота от шага $t-1$ к шагу t) и облако точек C_t, снятое лидаром робота на шаге t. Схема процесса обновления графа локаций изображена на рисунке 3.7.

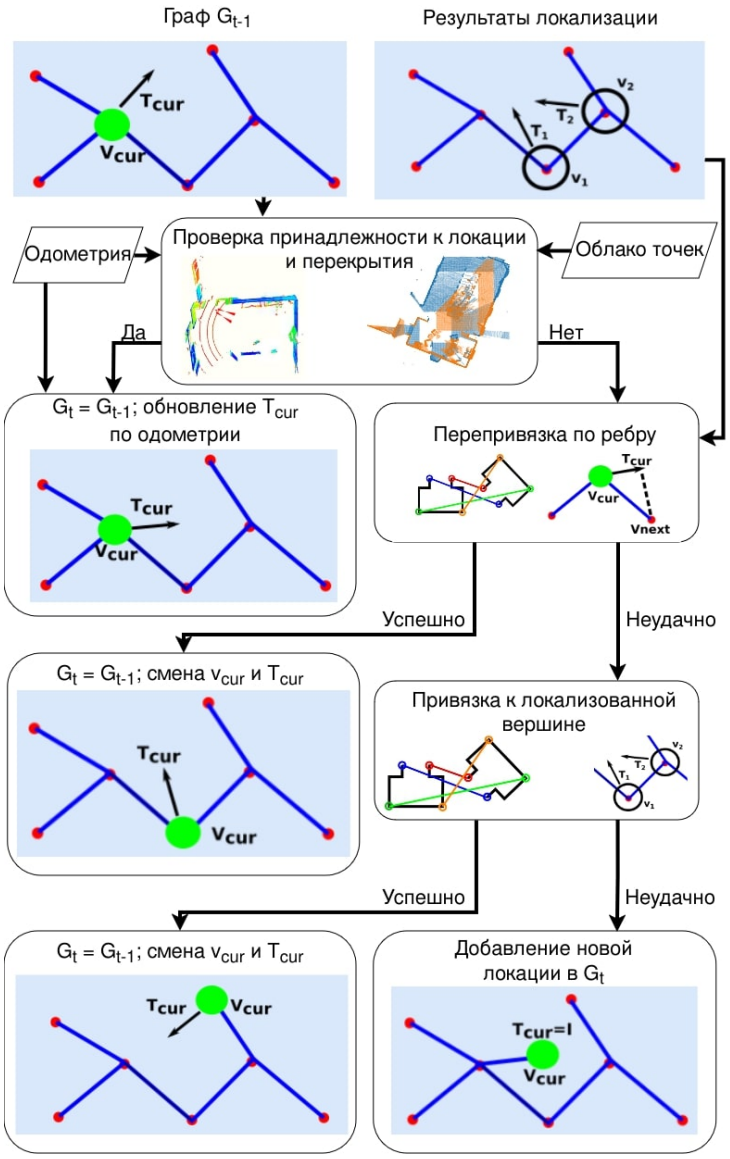

Рисунок 3.7 — Схема модуля поддержания графа: проверка принадлежности робота к локации v_{cur}^{t-1}, смена v_{cur} по ребру и по результатам локализации, добавление новой локации в граф.

Процедура обновления графа описана в алгоритме 4. В начале проводится проверка нахождения робота внутри локации v_{cur}^{t-1} и перекрытия текущего скана робота и скана локации v_{cur}^{t-1}. Если значение $overlap$ больше порога $t_{overlap}$ и робот находится внутри локации v_{cur}^{t-1}, то текущее состояние в графе обновляется исходя из одометрии.

Алгоритм 4: Процедура обновления графа на шаге t

Вход: Граф локаций $G_{t-1} = (V_{t-1}, E_{t-1})$; состояние робота в графе $(v_{cur}^{t-1} \in V_{t-1}, T_{cur}^{t-1} \in SE(3))$; облако точек $C_t \in \mathbb{R}^{N\times 3}$; данные одометрии $\widehat{\Delta S_t}$; результаты локализации: $\{(v_{loc}^i \in V_{t-1}, T_{loc}^i \in SE(3), matchScore^i \in [0,1])\}_{i=1,\dots,k}$, упорядоченные по убыванию $matchScore^i$

Выход: Обновленный граф $G_t = (V_t, E_t)$; обновленное состояние робота в графе (v_{cur}^t, T_{cur}^t)

Параметры: Порог доли перекрытия с текущей локацией $t_{overlap}$; порог качества сопоставления t_{match}

`/* 1. Проверка нахождения робота внутри локации` v_{cur}^{t-1} `и перекрытия сканов */`

if $IoU(T_{cur}^{t-1} C_{v_{cur}^{t-1}}, C_t) \geqslant t_{overlap}$ ***and*** $isInside(T_{cur}^{t-1}, C_{v_{cur}^{t-1}})$ **then**

  $v_{cur}^t = v_{cur}^{t-1};\ T_{cur}^t = T_{cur}^{t-1} \oplus \widehat{\Delta S_t};\ V_t = V_{t-1};\ E_t = E_{t-1}$

`/* 2. Попытка смены` v_{cur} `на одну из соседних локаций */`

else

  $changed = False$

  $V_{neighbor} = \{u \in V_{t-1} :\ (v_{cur}^{t-1}, u) \in E_{t-1}\}$

  for $v \in v_{neighbor}$ **do**

    $e = (v_{cur}^{t-1}, v)$

    if $||F(e)|| < ||T_{cur}^{t-1}||$ **then**

      $scan_t = Projection(C_t)$

      $T_{edge}, matchScore = ScanMatching(scan_v, scan_t)$

      if $matchScore > t_{match}$ **then**

        $changed = True$

        $v_{cur}^t = v;\ T_{cur}^t = T_{edge};\ V_t = V_{t-1};\ E_t = E_{t-1}$

        break

`/* 3. Попытка смены` v_{cur} `по результатам локализации */`

if ***not*** $changed$ **then**

  for $i = 1, \dots, k$ **do**

    if $matchScore_i > t_{match}$ ***and*** $IoU(C_{v_{loc}^i}, T_{loc}^i C_t) \geqslant t_{overlap}$ **then**

      $v_{cur}^t = v_{loc}^i;\ T_{cur}^t = T_{loc}^i;\ V_t = V_{t-1};\ E_t = E_{t-1} \cup (v_{cur}^{t-1}, v_{cur}^t)$

      $changed = True$

      break

`/* 4. Добавление новой локации в граф и смена` v_{cur} `на нее */`

if ***not*** $changed$ **then**

  $v_{cur}^t = newNode(C_t);\ T_{cur}^t = I$

  $V_t = V_{t-1} \cup \{v_{cur}^t\};\ E_t = E_{t-1} \cup (v_{cur}^{t-1}, v_{cur}^t)$

Если робот находится вне локации v_{cur}^{t-1} или значение $overlap$ ниже порога, то проводится попытка перейти по ребру графа (сменить локацию v_{cur} на одну из соседних с локацией v_{cur}^{t-1}). Для этого текущий скан с робота сопоставляется со сканами локаций – соседей v_{cur}^{t-1} с использованием детектора углов Харриса [88]. В качестве начального предположения для сопоставления используется относительная позиция, вычисленная исходя из T_{cur}^{t-1} и относительных позиций, записанных на ребрах графа. Если сканы некоторых соседних локаций успешно сопоставились с текущим сканом с робота, то в качестве локации для перехода берется локация v_{next}, имеющая ближайшую к текущей позиции робота точку наблюдения.

Если не удалось сопоставить текущее наблюдение с робота с соседними с v_{cur}^{t-1} локациями, то проводится попытка обновить v_{cur} по результатам локализации. Для этого для каждой локации $v_{loc} \in V_{loc}$ проверяется ее доля перекрытия с текущим сканом с робота. Среди всех локаций, у которых значение перекрытия превышает порог $t_{overlap}$, для перехода выбирается локация v_{loc} с наибольшим значением $matchScore$. Далее обновляется текущее состояние в графе, и в граф добавляется новое ребро из v_{cur}^{t-1} в v_{loc}.

Если подходящая локация для смены v_{cur} не найдена, то в граф добавляется новая локация v_{new}, соответствующая текущему наблюдению с робота. Добавленная локация соединяется ребром с локацией v_{cur}^{t-1}.

Таким образом, при каждом добавлении новой локации в граф она соединяется с предыдущими локациями, что гарантирует связность графа. При смене v_{cur} по результатам локализации (пункт (3) процедуры) автоматически происходит замыкание цикла в графе локаций. При этом замыкание цикла не требует проведения ресурсоемкой глобальной оптимизации, так как в предложенном алгоритме не требуется метрическая согласованность циклов.

3.4 Выводы по главе

В данной главе описан разработанный в ходе диссертационного исследования алгоритм PRISM-TopoMap, решающий поставленную в разделе 2.3 задачу топологического картирования и локализации. Разработанный алгоритм состоит из выполняемых параллельно процедур поддержания и обновления то-

пологической карты (графа локаций) и локализации в построенной карте. В каждый момент времени PRISM-TopoMap поддерживает текущее состояние в графе, состоящее из локации, в которой находится робот в данный момент, и относительного положения робота в этой локации.

Для локализации в топологической карте используется двухэтапный подход, состоящий из глобального поиска локации нейросетевыми методами распознавания места и фильтрации ложных распознаваний и уточнения положения робота с помощью оригинального алгоритма сопоставления двумерных проекций облаков точек. Так обеспечивается надежная локализация в топологической карте и исчезает необходимость хранить в карте облака точек локаций. При обновлении карты не используются глобальные метрические координаты и глобальная метрическая оптимизация, тем самым исключается накопление ошибки позиционирования и снижаются вычислительные затраты при картировании. Замыкание циклов происходит без использования ресурсоемких методов оптимизации при переходе в локацию из результатов процедуры локализации.

Глава 4. Программный комплекс топологического картирования и локализации

Разработанный в ходе диссертационного исследования алгоритм построения и поддержания графа локаций и локализации в построенном графе PRISM-TopoMap реализован в виде комплекса программных средств. Код программного комплекса написан на языке программирования Python и интегрирован с операционной системой роботов ROS [1], являющейся основным стандартом программирования в робототехническом сообществе. Реализация алгоритма PRISM-TopoMap доступна по ссылке [2]. Реализация составных частей процедуры локализации (распознавания места и сопоставления сканов) доступна по ссылке [3] в составе программной библиотеки OpenPlaceRecognition.

Программный комплекс позволяет изменять параметры алгоритма PRISM-TopoMap с помощью конфигурационного файла, сохранять построенный граф локаций на диск, загружать с диска и использовать при локализации сохраненный ранее граф. Разработанный программный комплекс позволяет оценивать качество сохраненного графа по формулам 2.28-2.31. Интеграция с системой ROS позволяет получать данные с бортовых датчиков различных робототехнических систем или из робототехнического симулятора без доработки исходного кода программного комплекса, а также обмениваться данными PRISM-TopoMap с другими программами, используемыми на роботе (постановка целей для навигации, управление роботом и др.).

4.1 Структура программного комплекса

Программный комплекс топологического картирования и локализации состоит из четырех основных классов:

[1] https://www.ros.org

[2] https://github.com/KirillMouraviev/PRISM-TopoMap

[3] https://github.com/OPR-Project/OpenPlaceRecognition/tree/feat/toposlam

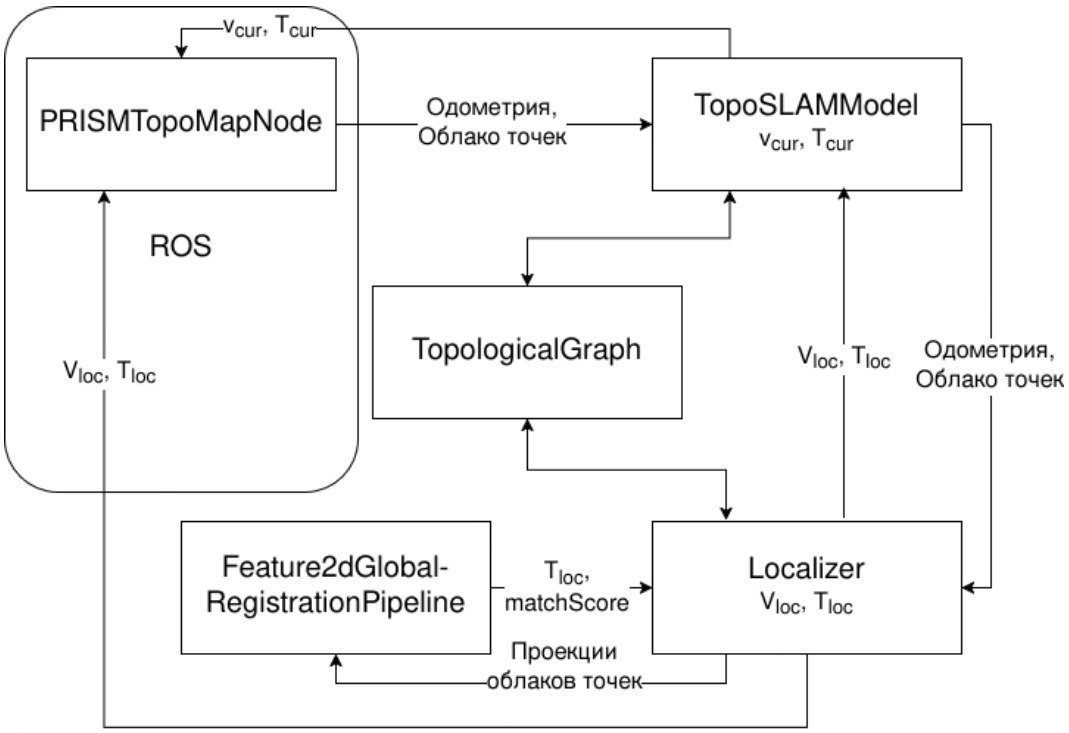

Рисунок 4.1 — Схема взаимодействия классов программного комплекса.

– **TopologicalGraph** – класс, реализующий граф локаций. Он осуществляет инициализацию графа с заданными параметрами, добавление в граф вершин и ребер, загузку графа с диска и его сохранение на диск.
– **Feature2DGlobalRegistrationPipeline** – класс, реализующий процедуру сопоставления сканов. Он осуществляет поиск преобразования координат между двумя сканами, представленными в виде проекций облаков точек, и вычисление качества найденного преобразования согласно алгоритму 3.
– **Localizer** – класс, реализующий процедуру локализации в графе локаций. Он принимает на вход экземпляры классов `TopologicalGraph` и `Feature2dGlobalRegistrationPipeline`, и по данным с бортовых датчиков робота выдает список локаций и положений робота относительно их точек наблюдения согласно алгоритму 1.
– **TopoSLAMModel** – класс, реализующий процедуру поддержания и обновления графа локаций. Он принимает на вход экземпляры классов `TopologicalGraph` и `Localizer`, и по данным одометрии, наблюдениям бортовых датчиков робота и результатам локализации обновляет граф и текущее состояние в нем согласно алгоритму 4.

Помимо вышеописанных классов, программный комплекс включает в себя класс **PRISMTopoMapNode**, реализующий взаимодействие классов алгорит-

ма PRISM-TopoMap с системой ROS, а также набор вспомогательных функций, реализующих геометрические операции, и класс **LocalGrid**, реализующий двумерную проекцию облака точек. Класс `PRISMTopoMapNode` осуществляет подачу на вход классам `TopoSLAMModel` и `Localizer` данных с робота (облака точек, изображения, одометрия) и отправку их выходных данных (граф локаций, текущее состояние в нем, результаты локализации) в ROS. Взаимодействие класса `PRISMTopoMapNode` с системой ROS осуществляется с помощью топиков [4] – очередей сообщений определенного типа, в которые могут публиковать и из которых могут читать сообщения различные программы, интегрированные в ROS. К примеру, топик `points` может являться очередью сообщений типа `sensor_msgs/PointCloud2` (облако точек), в которую публикует сообщения драйвер лидара и из которой читает сообщения класс PRISMTopoMapNode.

Для оценки качества топологического картирования по формулам 2.28-2.31 в составе программного комплекса реализован класс **MetricCounter**. Класс принимает на вход точную карту среды в виде сетки занятости. Его методы для подсчета путевой эффективности, количества компонент связности, корректности и полноты графа принимают на вход граф локаций в виде списка ребер. Описание атрибутов и методов классов `TopologicalGraph`, `Localizer`, `Feature2dGlobalRegistrationPipeline`, `TopoSLAMModel` и `MetricCounter` представлены ниже. Схема взаимодействия классов представлена на рисунке 4.1.

TopologicalGraph Атрибуты класса TopologicalGraph представлены в таблице 3. Методы класса TopologicalGraph описаны в таблице 4.

Таблица 3 — Атрибуты класса TopologicalGraph

vertices	List[Dict]	Список локаций – вершин графа
adj_lists	List[List]	Список смежности, в котором хранятся ребра графа
index	Faiss.Index	Экземпляр структуры для быстрого поиска ближайших дескрипторов с помощью библиотеки Faiss [5]

[4] https://wiki.ros.org/Topics

[5] https://github.com/facebookresearch/faiss

Таблица 4 — Методы класса TopologicalGraph

Метод	Входные параметры	Выходные параметры	Описание
__init__	Экземпляр класса, индекс для поиска ближайших дескрипторов, порог сопоставления сканов t_{match}	–	Инициализирует граф локаций с заданным индексом и порогом сопоставления
load_from_json	Экземпляр класса, путь к директории с сохраненным графом локаций	–	Загружает граф локаций из указанной директории
save_to_json	Экземпляр класса, путь к директории для сохранения графа локаций	–	Сохраняет построенный граф локаций в указанную директорию
add_vertex	Экземпляр класса, дескриптор для поиска, проекция облака точек (экземпляр класса `LocalGrid`), глобальная позиция для визуализации	Номер добавленной локации	Добавляет локацию в граф и возвращает ее порядковый номер в графе
get_vertex	Экземпляр класса, номер локации	Дескриптор, проекция и глобальная позиция для визуализации локации с заданным номером	Возвращает все атрибуты локации с заданным номером

Продолжение на следующей странице

Продолжение таблицы 4

Метод	Входные параметры	Выходные параметры	Описание
add_edge	Экземпляр класса, номера локаций, соединяемых ребром, позиция второй локации относительно первой	–	Добавляет в граф двустороннее ребро между парой локаций с заданной относительной позицией
has_edge	Экземпляр класса, номера двух локаций	Наличие ребра между двумя заданными локациями	Возвращает `True`, если в графе есть ребро между локациями, и `False` в противном случае
get_edge	Экземпляр класса, номера двух локаций	Позиция второй локации относительно первой, записанная на ребре	Возвращает относительную позицию в случае наличия ребра, и `None` в противном случае
get_transform_to_vertex	Экземпляр класса, номер локации, проекция облака точек с робота (экземпляр класса LocalGrid)	Преобразование координат от позиции робота до точки наблюдения указанной локации	Сопоставляет поданную на вход проекцию облака точек с робота с проекцией облака точек заданной локации. Возвращает найденное преобразование координат в случае его нахождения, в противном случае возвращает `None`
get_path_with_length	Экземпляр класса, номера двух локаций	Путь между двумя указанными локациями в графе, длина пути в метрах	Ищет путь между двумя указанными локациями в графе с помощью алгоритма Дейкстры. Возвращает путь в виде последовательности номеров локаций и примерную длину пути, вычисленную по записанным на ребрах относительным позициям

Feature2dGlobalRegistrationPipeline Атрибуты класса Feature2dGlobalRegistrationPipeline представлены в таблице 5. Методы класса Feature2dGlobalRegistrationPipeline описаны в таблице 6.

Таблица 5 — Атрибуты класса Feature2dGlobalRegistrationPipeline

Атрибут	Тип	Описание
detector	cv2.ORB | cv2.SIFT	Детектор особых точек, используемых для сопоставления
outlier_thresholds	List[Int]	Список пороговых значений для удаления выбросов ($\delta_1, \ldots, \delta_{max\_iter}$ из алгоритма 2)
min_matches	Int	Минимальное число сопоставлений особых точек (параметр K из алгоритма 2)

Таблица 6 — Методы класса Feature2dGlobalRegistrationPipeline

Метод	Входные параметры	Выходные параметры	Описание
__init__	Экземпляр класса, тип детектора особых точек (SIFT или ORB), количество особых точек, пороги сопоставления δ_{iter}, минимальное количество сопоставлений K	–	Инициализирует класс сопоставления облаков точек с заданными параметрами
get_fitness	Экземпляр класса, проекции двух облаков точек в виде сеток занятости, преобразование координат	Значение *matchScore* – качество сопоставления двух проекций облаков точек с помощью заданного преобразования	Вычисляет значение *matchScore* по заданным проекциям облаков точек и преобразованию координат согласно алгоритму 3

Продолжение таблицы 6

Метод	Входные параметры	Выходные параметры	Описание
infer	Экземпляр класса, проекции двух облаков точек в виде сеток занятости	Найденное преобразование координат между двумя облаками точек, а также значение *matchScore*	Выполняет процедуру сопоставления двух сканов (проекций облаков точек) с удалением выбросов согласно алгоритму 2. Возвращает найденное преобразование координат и его значение *matchScore*, в случае ненахождения преобразования возвращает `None`

Localizer Атрибуты класса Localizer представлены в таблице 7. Методы класса Localizer описаны в таблице 8.

Таблица 7 — Атрибуты класса Localizer

Атрибут	Тип	Описание
graph	TopologicalGraph	Граф локаций, в котором проводится локализация
top_k	Int	Количество наиболее близких по дескрипторам локаций k, возвращаемое на первом этапе локализации
registration_score_threshold	Float	Значение порога t_{match}, используемое на шаге 3 алгоритма 4
place_recognition_model	torch.nn.Model	Нейросетевая модель распознавания места, используемая на первом этапе локализации
scan_matching_pipeline	Feature2DGlobal-RegistrationPipeline	Экземпляр класса сопоставления сканов, используемый на втором этапе локализации

Таблица 8 — Методы класса Localizer

Метод	Входные параметры	Выходные параметры	Описание

Продолжение на следующей странице

Продолжение таблицы 8

Метод	Входные параметры	Выходные параметры	Описание
__init__	Экземпляр класса, граф локаций, нейросетевая модель распознавания места и алгоритм сопоставления сканов, значение top_k	–	Инициализирует класс локализации с заданными параметрами
localize	Экземпляр класса, облако точек с робота, изображения (опционально)	Номера локаций графа, в которых находится робот, и позиция робота относительно их точек наблюдения	Выполняет процедуру локализации в графе по поданному на вход наблюдению точек с робота

TopoSLAMModel Атрибуты класса TopoSLAMModel представлены в таблице 9. Методы класса TopoSLAMModel описаны в таблице 10.

Таблица 9 — Атрибуты класса TopoSLAMModel

Атрибут	Тип	Описание
graph	TopologicalGraph	Используемый экземпляр графа локаций
localizer	Localizer	Используемый экземпляр класса локализации
path_to_load_graph	Str	Путь для загрузки графа локаций с диска
path_to_save_graph	Str	Путь для сохранения построенного графа локаций на диск
last_vertex	(Array, LocalGrid, Array)	Текущая локация v_{cur}, хранимая как кортеж из дескриптора локации, проекции ее облака точек и, опционально, позиции ее точки наблюдения в глобальной системе координат
last_vertex_id	Int	Номер текущей локации v_{cur}
rel_pose_of_vcur	Array	T_{cur} – положение робота относительно точки наблюдения локации v_{cur}

Продолжение на следующей странице

Продолжение таблицы 9

Атрибут	Тип	Описание
iou_threshold	Float	Значение порога $t_{overlap}$ в алгоритме 4
max_edge_length	Float	Максимальная длина ребра, добавляемого в граф (при выезде за пределы этого расстояния от точки наблюдения локации v_{cur} происходит переход к шагу 2 алгоритма 4)
floor_height	Float	Максимальная высота точек поверхности z_{floor}, используемая при проекции облака точек
ceil_height	Float	Минимальная высота точек потолка z_{ceil}, используемая при проекции облака точек
inline_registration_model	Feature2DGlobalRegistrationPipeline	Экземпляр класса сопоставления сканов, используемый на шаге 2 алгоритма 4
inline_registration_score_threshold	Float	Значение порога t_{match}, используемое на шаге 2 алгоритма 4

Таблица 10 — Методы класса TopoSLAMModel

Метод	Входные параметры	Выходные параметры	Описание
__init__	Экземпляр класса, конфигурационный файл с параметрами, пути для загрузки и сохранения графа локаций	–	Инициализирует все составляющие алгоритма PRISM-TopoMap с параметрами из конфигурационного файла и заданными путями для загрузки и сохранения графа локаций
update	Экземпляр класса, облако точек с робота, одометрия, изображения с робота (опционально)	–	Выполняет процедуру обновления графа в соответствии с алгоритмом 4

Продолжение на следующей странице

Продолжение таблицы 10

Метод	Входные параметры	Выходные параметры	Описание
reattach_by_edge	Экземпляр класса, проекция облака точек с робота	Успешность перехода по ребру	Выполняет шаг 2 процедуры обновления графа и возвращает `True` в случае успешной смены v_{cur}, и `False` в противном случае
reattach_by_localization	Экземпляр класса	Успешность смены v_{cur}	Выполняет шаг 3 процедуры обновления графа и возвращает `True` в случае успешной смены v_{cur}, и `False` в противном случае
add_new_vertex	Экземпляр класса, дескриптор и проекция облака точек с робота	–	Выполняет шаг 4 процедуры обновления графа

MetricCounter Атрибуты класса MetricCounter представлены в таблице 11. Методы класса TopologicalGraph описаны в таблице 12.

Таблица 11 — Атрибуты класса MetricCounter

Атрибут	Тип	Описание
gt_map	Array	Точная карта среды в виде сетки занятости

Таблица 12 — Методы класса MetricCounter

Метод	Входные параметры	Выходные параметры	Описание
__init__	Экземпляр класса, путь к точной карте среды	–	Загружает точную карту среды
n_components	Экземпляр класса, список ребер графа локаций в координатах загруженной точной карты среды	Количество компонент связности	Вычисляет количество компонент связности графа

Продолжение на следующей странице

Продолжение таблицы 12

Метод	Входные параметры	Выходные параметры	Описание
area_covered_by_main_component	Экземпляр класса, список ребер графа локаций в координатах загруженной точной карты среды	Доля покрытия сцены	Вычисляет долю покрытия сцены локациями графа 2.29
edge_correctness	Экземпляр класса, список ребер графа локаций в координатах загруженной точной карты среды	Значение корректности ребер графа	Вычисляет корректность ребер графа по формуле 2.30
edge_recall	Экземпляр класса, список ребер графа локаций в координатах загруженной точной карты среды	Значение полноты ребер графа	Вычисляет полноту ребер графа по формуле 2.31
path_efficiency	Экземпляр класса, список ребер графа локаций в координатах загруженной точной карты среды	Значение путевой эффективности	Вычисляет путевую эффективность (SPL_N) по формуле 2.28

4.2 Параметры

Параметры алгоритма PRISM-TopoMap, а также класса PRISMTopoMapNode, реализующего взаимодействие алгоритма PRISM-TopoMap с системой ROS, задаются в конфигурационном файле формата `yaml`. Путь к конфигурационному файлу, пути для загрузки и/или сохранения графа локаций прописываются в файле запуска ROS формата `launch`. Описание параметров и их значения, использованные в экспериментальном исследовании в симуляционных помещениях, представлены в таблице 13.

Таблица 13 — Значения параметров алгоритма PRISM-TopoMap, используемых для симуляционных помещений

Класс	Параметр	Описание	Значение
TopoSLAM-Model	floor_height	Максимальная высота точек поверхности z_{floor} в формуле 3.8	-0.9
	ceil_height	Минимальная высота точек потолка z_{ceil} в формуле 3.8	1.5
	iou_threshold	Значение порога $t_{overlap}$ на шаге 1 алгоритма 4	0.3
	max_edge_length	Максимальная длина ребра, добавляемого в граф	5.0
	inline_registration_score_threshold	Значение порога t_{match}, используемое на шаге 2 алгоритма 4	0.5
Localizer	place_recognition_model	Тип нейросетевой модели распознавания места	MSSPlace
	registration_score_threshold	Значение порога t_{match}, используемое на шаге 3 алгоритма 4	0.6
	top_k	Количество наиболее близких по дескрипторам локаций k, возвращаемое на первом этапе локализации	5
Feature2D-Global-Registration-Pipeline	detector_type	Тип детектора особых точек, используемого при сопоставлении сканов	ORB
	outlier_thresholds	Список пороговых значений для удаления выбросов $(\delta_1, \ldots, \delta_{max\_iter}$ из алгоритма 2) в метрах	(2.5, 1, 0.5, 0.25, 0.25)
	min_matches	Минимальное число сопоставлений особых точек (параметр K из алгоритма 2)	5
LocalGrid	resolution	Размер ячейки двумерной проекции облака точек (параметр d в формуле 3.8) в метрах	0.1

Продолжение на следующей странице

Продолжение таблицы 13

Класс	Параметр	Описание	Значение
	max_range	Максимальное расстояние от точки наблюдения при проекции (параметр r в формуле 3.8) в метрах	8.0

Значения параметров `floor_height` и `ceil_height` подбираются исходя из высоты лидара робота над полом, а также высоты потолка (в случае работы в помещениях). К примеру, указанные в таблице значения подходят для высоты лидара робота над полом 1 м и для высоты потолка 3 м. В таком случае пол имеет z-координату, равную -1.0, а потолок – равную 2.0, и в проекцию попадают все точки, которые выше пола более чем на 0.1 м и ниже потолка более чем на 0.5 м. Для использования PRISM-TopoMap на улице целесообразно немного повысить значение параметра `floor_height` для компенсации неровностей поверхности. Значение `ceil_height` при использовании на улице можно выставить большим, например, 5.0.

Значение параметра `iou_threshold` определяет частоту перехода к шагам 2-4 алгоритма 4 и, соответственно, плотность графа локаций. Чем выше это значение, тем чаще будет выполняться переход и/или добавление новой локации, и тем плотнее будет граф. Оптимальными являются значения в диапазоне от 0.2 до 0.4. Значение параметра `max_edge_length` тоже влияет на плотность графа – чем меньше максимальная длина добавляемого в граф ребра, тем плотнее получится граф.

Выбор значений параметров `registration_score_threshold` и `inline_registration_score_threshold` определяется плотностью облака точек с робота и объектов в среде, а также интенсивностью динамических изменений среды. При разреженном облаке точек или низкой плотности объектов в среде (в частности, при работе на улице), или при значительных изменениях среды следует выбирать меньшие значения параметров, чтобы сопоставления успешно находились. При высокой плотности облака точек и/или объектов и незначительных изменениях среды (работа в помещениях без людей) следует выбирать более высокие значения параметров для более точного сопоставления.

Значение параметра `min_matches` менее 4 может привести к некорректному сопоставлению, более 7 – к отбрасыванию корректных сопоставлений.

Таким образом, рекомендуется выбирать `min_matches` = 5. Значения параметра `outlier_thresholds` подобрано эмпирическим путем в ходе экспериментов в симуляционных помещениях. Оптимальный результат достигается при значениях последнего элемента из списка `outlier_thresholds` в пределах 2-3 размеров ячейки двумерной проекции ($2 \cdot resolution < \delta_{max\_iter} < 3 \cdot resolution$).

Значение размера ячейки проекции `resolution` определяется желаемой точностью локализации (оно должно быть меньше, чем желаемое значение ошибки), а также требованиями к компактности топологической карты (ее размер в памяти обратно пропорционален квадрату значения `resolution`). В помещениях оптимальное значение порядка 0.1, на улице – порядка 0.25. Значение радиуса проекции `max_range` определяется типичным расстоянием от робота до объектов, попадающих в его наблюдение (как правило, в помещениях оно равно 5-15 м, на улице – 20-50 м).

Описание входных и выходных топиков ROS представлено в таблице 14. На вход классу `PRISMTopoMapNode` в соответствующих топиках подаются все входные данные алгоритма PRISM-TopoMap: одометрия, облако точек, изображения с передней и задней камер. Все основные выходные данные публикуются в соответствующие топики ROS: граф локаций, текущая локация v_{cur}, положение в ней T_{cur}, проекция текущего скана с робота и локации v_{cur}, сопоставленные и не сопоставленные процедурой локализации локации.

Таблица 14 — Топики для получения и отправки данных в систему ROS классом PRISMTopoMapNode

	Топик	Тип	Описание
Вход	odom	nav_msgs/ Odometry	Данные одометрии
	points	sensor_msgs/ PointCloud2	Облако точек с робота
	image_front	sensor_msgs/ Image	Изображение с передней камеры робота
	image_back	sensor_msgs/ Image	Изображение с задней камеры робота
Выход	local_grid	nav_msgs/ OccupancyGrid	Проекция скана локации v_{cur}
	last_vertex	visualization_msgs/ Marker	Точка наблюдения локации v_{cur}

Продолжение на следующей странице

Продолжение таблицы 14

	Топик	Тип	Описание
	topological_ map	visualization_msgs/ MarkerArray	Граф локаций (визуализируются точки наблюдения локаций и ребра)
	matched_ points	visualization_msgs/ Marker	Точки наблюдения локаций, успешно сопоставленных процедурой локализации
	unmatched_ points	visualization_msgs/ Marker	Точки наблюдения локаций, входящих в k ближайших в пространстве дескрипторов, но не сопоставленных процедурой сопоставления сканов
	current_grid	nav_msgs/ OccupancyGrid	Проекция текущего облака точек с робота
	tf	tf2_msgs/ TFMessage	Преобразования координат ($map \rightarrow vcur$ – от глобальной системы координат до системы координат, связанной с точкой наблюдения v_{cur}, если задана глобальная метрическая система координат, и $vcur \rightarrow current\_state$ – преобразование T_{cur})

4.3 Пример использования

В данном разделе показан пример запуска разработанного программного комплекса картирования и локализации на симуляционном помещении в симуляторе Habitat. Пример повторяет эксперимент, проведенный на первой из пяти сцен в разделе 5.2. Для запуска симулятора Habitat и подачи симулированных наблюдений с робота в систему ROS используется программный пакет `habitat_ros`[6]. Посредством команды `roslaunch` пакет запускает симулятор Habitat на заданной сцене с моделированием указанных в эксперименте датчиков и подает наблюдения с робота (облако точек и четыре RGB-D изоб-

[6] https://github.com/CnnDepth/habitat_ros/tree/toposlam_experiments

ражения: вперед, назад, влево и вправо) в систему ROS. Команда для запуска пакета `habitat_ros` выглядит так:

```
roslaunch habitat_ros toposlam_experiment_mp3d_4x90_large_noise.launch
```

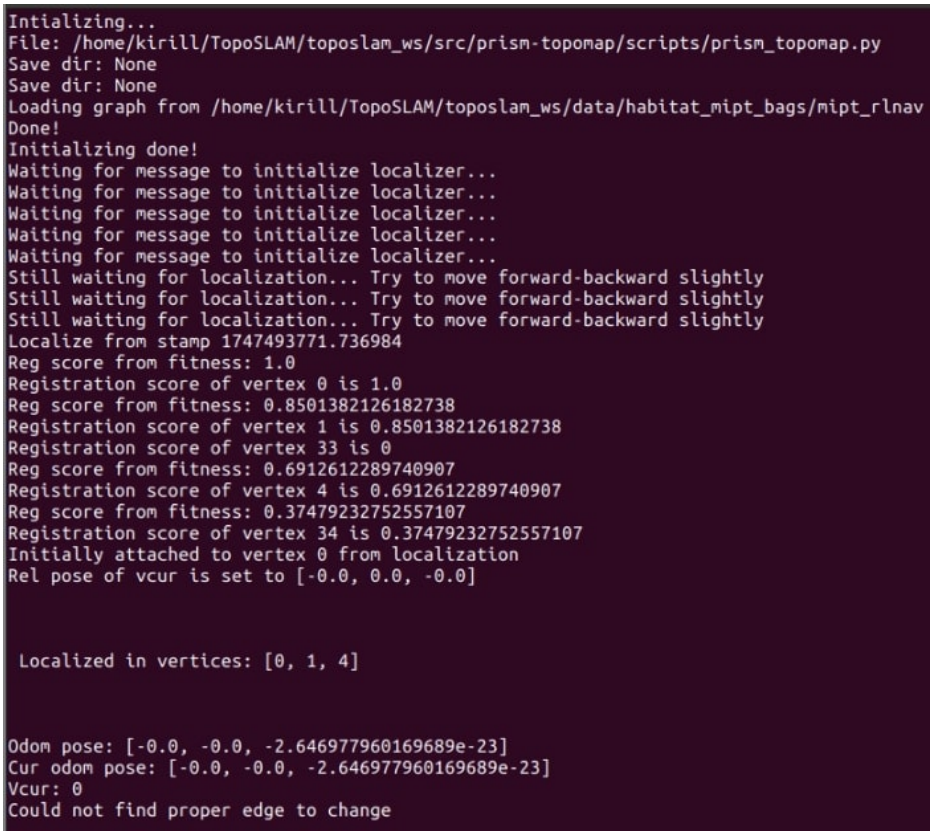

Рисунок 4.2 — Пример вывода программного комплекса в консоль.

Далее пользователю нужно создать конфигурационный файл, в котором будут прописаны значения параметров алгоритма PRISM-TopoMap, а также названия входных топиков ROS. Конфигурационный файл для экспериментов в помещениях в симуляторе Habitat доступен по ссылке [7]. Далее путь к конфигурационному файлу и пути для загрузки и/или сохранения графа локаций прописываются в файл для запуска программного комплекса в системе ROS. Пример такого файла представлен по ссылке [8]. Запускается программный комплекс следующей командой:

```
roslaunch prism_topomap build_map_by_iou_habitat.launch
```

После запуска программный комплекс выводит информацию о процессе картирования и локализации в командную консоль. Пример вывода представлен на рисунке 4.2. Наблюдения с робота, граф локаций и текущее состояние

[7] https://github.com/KirillMouraviev/PRISM-TopoMap/blob/main/config/habitat_mp3d_noised_odom.yaml
[8] https://github.com/KirillMouraviev/PRISM-TopoMap/blob/main/launch/build_map_by_iou_habitat.launch

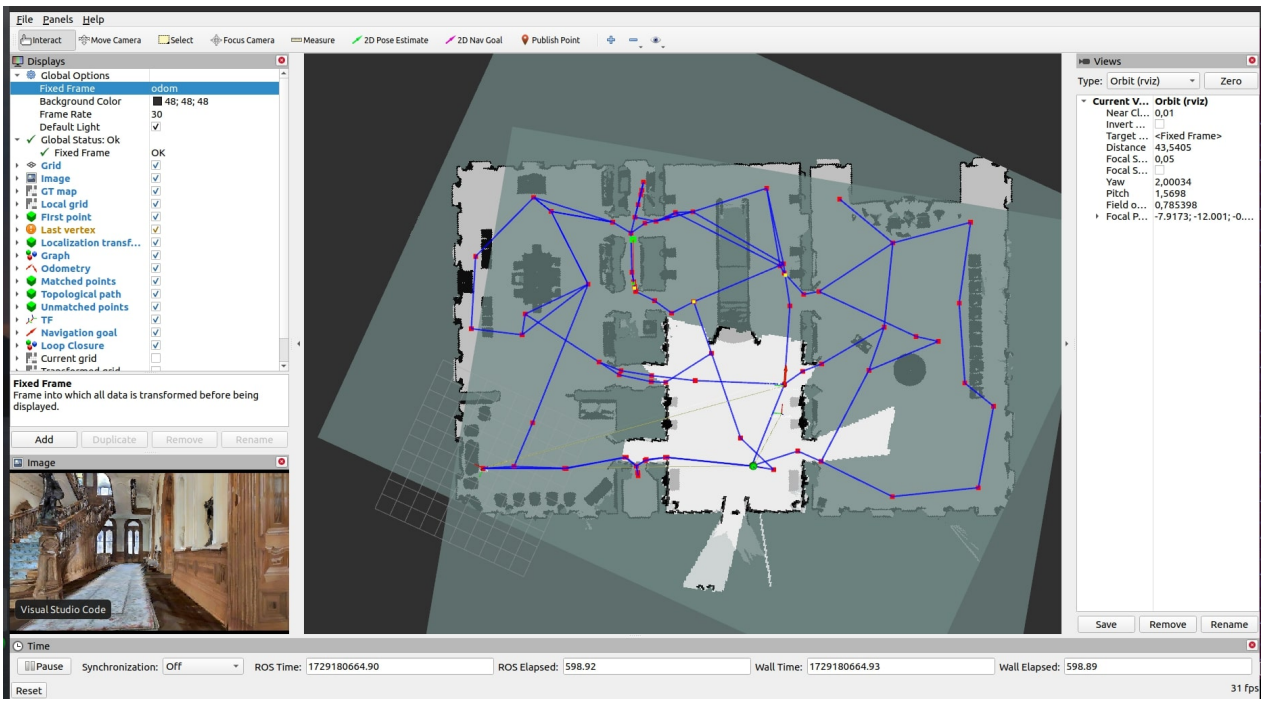

Рисунок 4.3 — Пример визуализации работы программного комплекса в RViz: граф локаций и текущее состояние в нем в сравнении с двумерной сеткой занятости, построенной по всей сцене.

робота в нем можно наблюдать с помощью `RViz` – стандартного инструмента визуализации для системы ROS. Конфигурационный файл формата `rviz` доступен по ссылке [9]. Пример визуализации графа локаций с помощью RViz представлен на рисунке 4.3. Граф локаций показан красными квадратами и синими линиями, точка наблюдения v_{cur} – зеленым кругом. Локации, найденные моделью распознавания места, которые не удалось сопоставить с текущим наблюдением с робота, показаны желтыми квадратами. Черно-белым показана проекция текущего облака точек с робота.

4.4 Выводы по главе

На основе разработанного алгоритма топологического картирования и локализации PRISM-TopoMap, описанного в предыдущей главе, был создан комплекс программных средств. Комплекс реализован на языке программирования Python и интегрирован с операционной системой роботов ROS, что позволяет

[9] https://github.com/CnnDepth/habitat_ros/blob/toposlam_experiments/rviz/toposlam_experiment_habitat.rviz

использовать его на различных робототехнических системах и в симуляторах без изменения исходного кода. С помощью данного программного комплекса пользователь может задавать подходящие под задачу параметры алгоритма PRISM-TopoMap в конфигурационном файле, сохранять и загружать построенные графы локаций.

Программный комплекс топологического картирования и локализации состоит из четырех основных классов, реализующих основные составляющие алгоритма PRISM-TopoMap (граф локаций, процедура сопоставления сканов, процедура локализации, процедура обновления графа), а также из нескольких вспомогательных (взаимодействие с системой ROS, проекция облака точек, геометрические преобразования, вычисление критериев качества по формулам 2.28-2.31). В главе приведены атрибуты и методы основных классов и значения параметров, используемых в симуляционных экспериментах в разделе 5.2. Приведен пример запуска программного комплекса для проведения симуляционного эксперимента. Исходный код всех классов комплекса выложен в открытый доступ [10] и зарегистрирован в виде программы для ЭВМ (см. приложение А).

[10] https://github.com/kirillMouraviev/PRISM-TopoMap

Глава 5. Экспериментальное исследование

Для оценки качества картирования и локализации с помощью предложенного набора критериев качества необходимо проведение численного эксперимента с построением карты некоторой среды, локализацией в карте и вычислением значений метрик. Для этой цели подходят робототехнические симуляторы, в которых строится точная модель среды и робота и моделируются наблюдения с бортовых датчиков робота. В данной главе описана постановка численного эксперимента в фотореалистичном симуляторе Habitat [89] с моделированием ошибки одометрии, встречающейся на реальном роботе. В ходе численного эксперимента проводится проезд робота по маршруту, наблюдения с которого охватывают всю среду, выполнение ОКЛ в реальном времени и оценка качества картирования и локализации с использованием всех предложенных в главе 2 показателей качества. Выполняется сравнение разработанного алгоритма PRISM-TopoMap с другими современными алгоритмами ОКЛ по показателям 2.28-2.31, а также по затратам времени и памяти в процессе картирования.

Помимо численных экспериментов в симуляторе, проводятся натурные эксперименты на данных с реальных робототехнических систем. Эксперименты состоят из картирования помещений большой площади на двух проездах мобильного робота в помещениях большой площади и локализации мобильного робота в предварительно построенной карте. Для локализации используется проезд робота по открытой среде длиной 3 км. Выполняется сравнение качества локализации по показателям 2.16, 2.34 с другими современными алгоритмами.

5.1 Постановка численного эксперимента в симуляционной среде

Рассматриваемые в данном исследовании алгоритмы топологического картирования и локализации предназначены для работы на реальных робототехнических системах в ходе их автономной навигации. Однако оценка качества в ходе тестирования на реальной робототехнической системе затруднительна по следующим причинам:

1. Значительные затраты времени на подготовку и проведение эксперимента;
2. Проблемы воспроизводимости эксперимента – повторить несколько раз эксперимент на реальной робототехнической системе с соблюдением одних и тех же условий затруднительно;
3. Трудности с получением истинных данных о положении робота и окружающих его объектов – отслеживание положения реальных объектов с высокой точностью требует наличия дорогостоящего оборудования и тщательной пост-обработки результатов.

Проблема воспроизводимости решается наличием открытых коллекций данных, содержащих проезды реальных робототехнических систем в помещениях и на улице (например, EuRoC [90], TUM RGB-D [91], KITTI [70], ITLP-Campus [8]). Помимо наблюдений с бортовых датчиков роботов (лидары, стереокамеры, RGB-D камеры, ИНС) эти коллекции содержат траектории роботов, измеренные с высокой точностью. Однако такие коллекции не содержат точную модель среды с положением всех объектов, попавших в наблюдения роботов, что сокращает возможности оценки качества картирования.

Для получения данных об истинном положении окружающих робототехническую систему объектов, а также для прогона робота по произвольной траектории используются различные робототехнические симуляторы. В симуляторах создается подробная геометрическая модель окружающей среды, в которой моделируется движение робота и наблюдения с его бортовых датчиков. Таким образом, в симуляционной среде можно оценивать качество алгоритма ОКЛ по показателям 2.16,2.18,2.28 на большом количестве проездов робота.

Одним из наиболее известных и применяемых в робототехническом сообществе является симулятор Gazebo [92]. Его основными преимуществами являются простота использования и низкие требования к вычислительным ресурсам – симулятор работает даже на маломощных вычислителях. Существенным недостатком симулятора Gazebo является однообразность текстур в помещениях, затрудняющая работу визуальных методов ОКЛ. Более того, в Gazebo некорректно моделируются некоторые физические процессы – например, свет может проходить сквозь непрозрачные объекты (см. рисунок 5.1).

Еще одним широко используемым робототехническим симулятором является ISAAC Sim [1]. В этом симуляторе более точно смоделированы физические

[1] https://developer.nvidia.com/isaac/sim

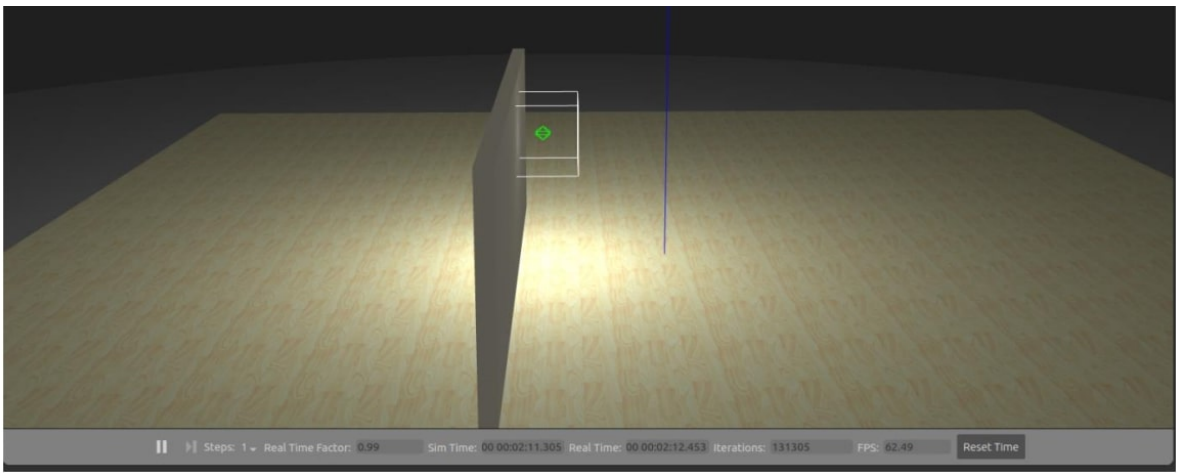

Рисунок 5.1 — Прохождение света сквозь стену в симуляторе Gazebo. Источник [82].

процессы (движение робота, распространение света и т.д.), а также имеются высокотекстурированные модели сред. ISAAC позволяет обучать различные нейросетевые методы взаимодействия с окружающей средой с помощью расширения ISAAC Gym [93]. Однако для запуска симулятора требуется наличие мощного графического ускорителя и не менее 8 ГБ видеопамяти, что доступно не на всех современных компьютерах.

Для проведения численных экспериментов в данной работе был выбран симулятор Habitat [89]. Он обладает высокой фотореалистичностью и имеет значительно более низкие требования к вычислительным ресурсам, чем ISAAC Sim, за счет отсутствия моделирования кинематики робота (время в симуляторе дискретное, и на каждом шаге робот перемещается телепортацией между точками свободного пространства среды). Таким образом, симулятор Habitat не требует наличия таких мощных графических ускорителей, как симулятор ISAAC, и при этом в нем можно вычислять все описанные выше метрики качества картирования и локализации. Примеры визуализации сред в симуляторах Gazebo, ISAAC и Habitat показаны на рисунке 5.2.

Окружающая среда в симуляторе Habitat моделируется с помощью трехмерной полигональной сетки, грани которой задают поверхности, описывающие границы области объектов W_{obj} (2.1). Каждая грань имеет лицевую сторону (обращенную в свободную область W_{free}) и тыльную сторону (обращенную в область объектов W_{obj}). Лицевая сторона каждой грани имеет цвет, отображающийся на наблюдениях с моделей камер робота. Сетка имеет высокое

Рисунок 5.2 — Пример визуализации среды и модели робота в различных робототехнических симуляторах: (а) – Gazebo, (б) – ISAAC Sim, (в) – Habitat (модели робота в симуляторе Habitat не предполатается).

разрешение (диаметр ячейки порядка 1 см) для обеспечения фотореалистичности.

Симулятор Habitat поддерживает моделирование монокулярных и RGB-D камер, а также предоставляет точную позицию робота $\mathbf{p_t}$ (2.3). Изображения с камер робота моделируются следующим образом: полигональная сетка, описывающая границу W_{obst}, проецируется на матрицу камеры. Цвет каждого пикселя изображения определяется как цвет грани полигональной сетки, чья проекция попала на этот пиксель. Путем проекции полигональной сетки на матрицу камеры строится точная карта глубин изображения.

Робот представляется в виде цилиндра радиусом r и высотой h с заданной ориентацией (в экспериментах использовалась высота $h = 0.88$ м и радиус $r = 0.18$ м). Положение камеры задается в центре верхней грани этого цилиндра. Положение робота в позиции p_t задается по центру нижней грани цилиндра. Движение робота моделируется дискретно – на каждом шаге симуляции робот может повернуться влево или вправо на угол α, проехать вперед на расстояние d, либо остаться на месте. При этом перемещение робота вперед из положения p_t возможно, если и только если его цилиндр в положении $S_t \cdot (d, 0, 0)^T$ не пересекается с поверхностью области препятствий, заданной полигональной сеткой.

Для получения наблюдений с робота в симуляторе Habitat в соответствии с форматом 2.7 облако точек строится путем обратной проекции глубин изображений с камер. Для создания панорамного облака точек моделируются 4 камеры с горизонтальным полем зрения 90° каждая, направленные вперед, вле-

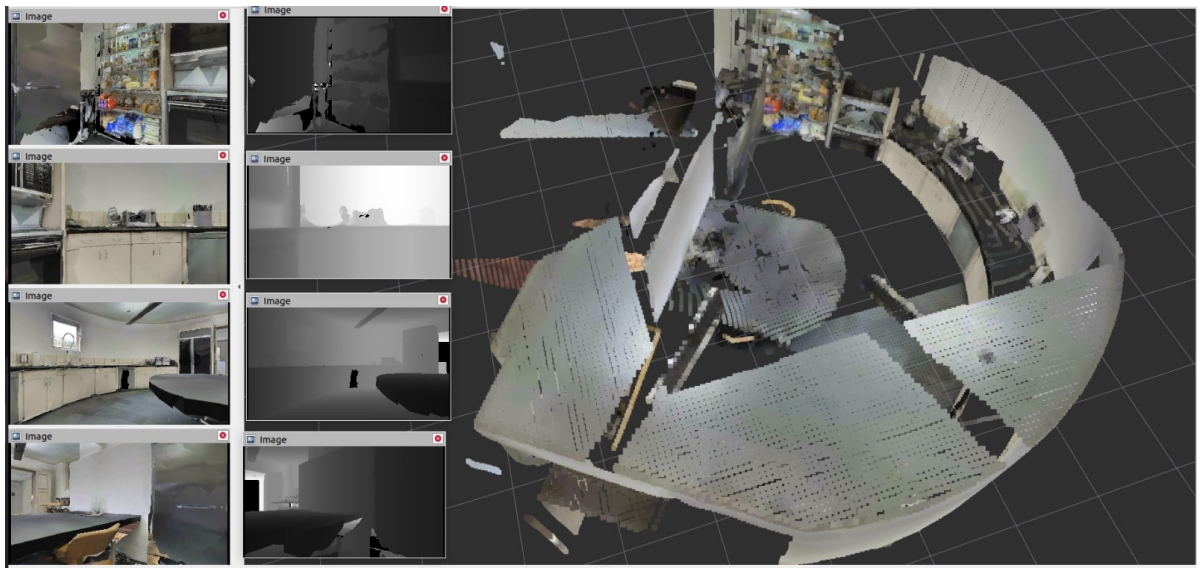

Рисунок 5.3 — Пример построения панорамного облака точек по изображениям с четырех RGB-D камер. Слева – изображения и глубины с камер, справа - облако точек, полученное с помощью обратной проекции.

во, вправо и назад. Пример создания облака точек по четырем RGB-D камерам прдедставлен на рисунке 5.3.

Для проверки алгоритмов ОКЛ на устойчивость к ошибке одометрии, которая неизбежно присутствует на реальных робототехнических системах, по точной позе робота S_t строится зашумленная позиция $\widehat{S}_t$ путем добавления гауссовского шума в соответствии с формулами 2.9, 2.13.

Симулятор Habitat не предоставляет функционал для получения трехмерной или двумерной модели помещения напрямую, поэтому для вычисления длин путей в формуле (2.26) строится срез модели помещения $W_{z_1:z_2}$ (2.2) и преобразуется в сетку занятости $grid_{z_1,z_2}$ с разрешением 0.05 м. Для вычисления среза по модели помещения строится трехмерное облако точек путем объезда помещения роботом и объединения локальных облаков точек, полученных с помощью обратных проекций глубин, в единое глобальное облако точек. Для объединения облаков точек используются истинные позиции робота p_t, полученные из симулятора. Для экономии памяти проводится дискретизация облака точек – все координаты округляются до 0.05 м, и после округления удаляются дубликаты точек. Далее берется срез построенного глобального облака точек по высотам от z_1 до z_2 и проецируется на плоскость. В экспериментах используются значения $z_1 = 0; z_2 = 1.0$, что соответствует высотам объектов, являющихся препятствиями для робота.

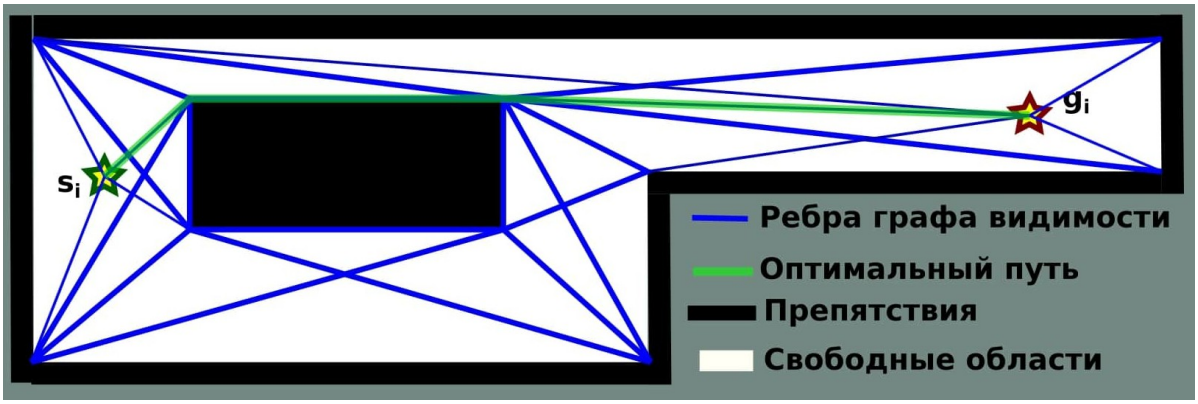

Рисунок 5.4 — Пример вычисления оптимального пути по сетке занятости с помощью графов видимости.

Для вычисления оптимальных путей по построенной сетке занятости проводится т.н. «раздутие препятствий» на радиус r: каждая ячейка сетки помечается занятой, если на расстоянии не более r от нее находятся занятые ячейки. После «раздутия препятствий» по сетке занятости строится граф видимости [94], в котором вершинами являются угловые точки препятствий, а ребрами соединяются угловые точки, отрезок между которыми лежит в свободной области среды W_{free}. Для построения графа видимости по сетке занятости используется реализация, описанная в работе [95]. При вычислении значения метрики $SPL(path(\mathbf{s_i}, \mathbf{g_i}, G))$ (2.28) в граф добавляются вершины, соответствующие точкам $\mathbf{s_i}, \mathbf{g_i}$. Построенные по таким графам пути являются оптимальными в сетке занятости $grid_{z_1:z_2}$ и, как следствие, являются близкими к оптимальному в срезе окружающей среды $W_{z_1:z_2}$. Пример вычисления пути по графам видимости показан на рисунке 5.4.

5.2 Численные эксперименты в симуляционной среде

Разработанный алгоритм топологического картирования и локализации в топологической карте был апробирован на симуляционных данных внутри помещений и на данных с реальных роботов внутри и вне помещений. Экспериментальное исследование включало в себя численный эксперимент в симуляционной среде в соответствии с постановкой, описанной в главе 2 с вычислением показателей качества 2.28-2.31, а также измерение количества памяти, потребляемой на хранение и поддержание карты, и времени, затра-

ченного на одну итерацию обновления карты. Было проведено сравнение с другими современными метрическими и топологическими методами картирования и локализации. Было проведено отдельное экспериментальное исследование алгоритмов сопоставления сканов для локализации в графе локаций на наборе симуляционных данных внутри помещений.

Исследование алгоритмов сопоставления сканов для локализации Экспериментальное исследование алгоритмов сопоставления сканов проводилось на наборе симуляционных данных внутри помещений. Набор содержал 5 сцен из коллекции данных Matterport3D [96]. По каждой сцене был построен маршрут, наблюдения с которого охватывали всю сцену, затем была построена топологическая карта этого маршрута и осуществлен проезд по маршруту с помощью симулятора Habitat [89]. На каждой точке маршрута бралось текущее наблюдение с робота, а также 5 локаций из топологической карты, чьи дескрипторы были наиболее близки к дескриптору текущего наблюдения. Таким образом, было получено 1605 поднаборов данных, каждый из которых содержал одно наблюдение с робота и 5 локаций-кандидатов, т.е. всего набор данных содержал 8025 пар сканов. Из них значение перекрытия (IoU) более 0.5 имело 1377 пар сканов, значение IoU более 0.25 – 2369 пар сканов.

Для сравнения были выбраны наиболее распространенные классические и нейросетевые методы сопоставления сканов: итеративный метод ближайшей точки (ICP) [73] с начальным приближением, получаемым с помощью алгоритма RANSAC [39], метод ICP с 6 разными случайно выбранными начальными приближениями и нейросетевая модель сопоставления сканов GeoTransformer [97]. Качество сопоставления оценивалось измерением точности и полноты сопоставления сканов. Точность измерялась как отношение числа корректно сопоставленных сканов к числу всех сканов, которые были сопоставлены алгоритмом. Полнота измерялась как отношение числа корректно сопоставленных сканов к числу всех сканов, которые имеют определенную долю перекрытия. Было проведено два измерения полноты – для всех пар сканов со значением IoU > 0.5 и для всех пар сканов со значением IoU > 0.25. Для каждого алгоритма было измерено среднее время, затраченное на сопоставление одной пары сканов. Результаты экспериментов приведены в таблице 15. Пример сопоставления сканов с помощью всех рассмотренных алгоритмов приведен на рисунке 5.5.

Таблица 15 — Результаты экспериментального исследования сопоставления сканов. Предложенный в данной работе алгоритм обозначен как Feature2D и опробован с двумя детекторами особых точек: SIFT и ORB

Алгоритм	Точность↑	Полнота > 0.5↑	Полнота > 0.25 ↑	Время, мс↓
RANSAC + ICP	0.84	0.75	0.53	360
6x ICP	0.78	0.09	0.05	580
Geotransformer	0.22	0.85	0.67	280
Feature2D-SIFT	**1.00**	0.85	0.56	75
Feature2D-ORB	**1.00**	**0.97**	**0.69**	**12**

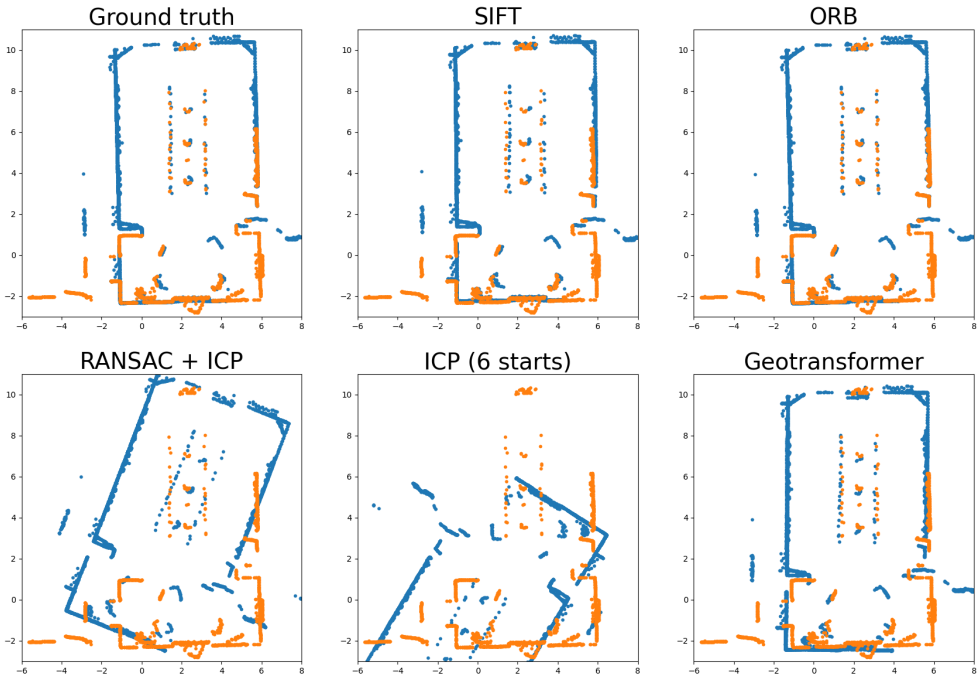

Рисунок 5.5 — Результаты сопоставления сканов для всех рассмотренных алгоритмов, в сравнении с истинным сопоставлением (Ground truth). Значение IoU у представленной пары сканов равно 0.65.

Исследование картирования и локализации в симуляционных помещениях Для экспериментального исследования картирования и локализации были выбраны 5 сцен из набора данных Matterport3D, использованных для исследования сопоставления сканов. По каждой из сцен был построен маршрут, наблюдения с точек которого охватывали всю сцену. Площадь сцен составляла от 100 до 700 м^2, длины маршрутов – от 100 до 300 м. По маршрутам был осуществлен проезд виртуального робота с помощью симулятора Habitat, в ходе которого с помощью алгоритма строилась карта с нуля.

На каждом шаге на вход алгоритму подавалось облако точек и изображения с камер робота, а также текущее положение робота из симулятора. Текущее

положение робота зашумлялось в соответствии с формулами 2.9, 2.13. Были использованы три уровня шума: нулевой, средний и большой. На среднем уровне шума были выбраны значения $\sigma_t^x = 0.003; \sigma_t^y = 0; \sigma_t^z = 0; \sigma_t^r = 0.0075$. На большом уровне шума – $\sigma_t^x = 0.0075; \sigma_t^y = 0; \sigma_t^z = 0; \sigma_t^r = 0.025$. Такие значения уровня шума соответствуют современным методам одометрии, относительная ошибка которых составляет, как правило, от 0.005 до 0.01. Для оценки качества алгоритма использовалась карта, построенная алгоритмом по завершении проезда по маршруту.

Для сравнения был выбран современный топологический алгоритм TSGM [46], два современных топометрических алгоритма Hydra [34] и IncrementalTopo [98], и три широко применимых метрических алгоритма: RTAB-Map [16], GLIM [13] и ORB-SLAM3 [10]. В качестве источника одометрии, корректирующего зашумленную одометрию из симулятора, для алгоритма RTAB-Map был выбран Cartographer [20]. На вход всем остальным алгоритмам, кроме TSGM, подавалась зашумленная одометрия из симулятора и облако точек. При этом алгоритмы ORB-SLAM3, GLIM и TSGM данные одометрии не использовали. На вход алгоритму TSGM подавалось панорамное RGB-D изображение из симулятора, разбитое на 12 секторов. Дополнительно на вход алгоритму Hydra подавались на вход данные семантической сегментации, вычисленные с помощью нейросетевой модели SegFormer [99].

Для оценки качества работы разработанного алгоритма и базовых алгоритмов использовались показатели 2.28-2.31: количество компонент связности (N_{comp}), доля покрытия сцены ($Coverage$), корректность и полнота ребер графа ($Correctness; Recall$) и путевая эффективность (SPL_N для $N = 1000$). Значения всех используемых показателей для всех рассмотренных алгоритмов при всех значениях шума позиции приведены в таблице 16. Построенные алгоритмами карты показаны на рисунке 5.6.

Как видно из таблицы, предложенный алгоритм PRISM-TopoMap при всех уровнях шума позиции построил связные графы с высоким покрытием сцены, по покрытию лишь незначительно уступая алгоритмам ORB-SLAM3 и TSGM. По значению SPL_N на нулевом и среднем шуме позиции PRISM-TopoMap незначительно уступил алгоритму RTAB-Map, при этом значительно обойдя RTAB-Map на высоком уровне шума (при высоком уровне шума позиции RTAB-Map построил несвязную карту на одной сцене). При этом у алгоритма PRISM-TopoMap значения метрик покрытия и SPL с ростом уровня шума уве-

Таблица 16 — Значение показателей качества 2.28-2.31 по результатам симуляционных экспериментов

Шум	Алгоритм	$N_{comp}\downarrow$	Coverage↑	Correctness↑	Recall↑	$SPL_N\uparrow$
Нулевой	RTAB-Map	**1.0**	0.79	–	–	**0.86**
	ORB-SLAM3	2.2	**0.95**	**1.00**	0.00	0.74
	GLIM	**1.0**	0.80	0.94	**0.27**	0.72
	Hydra	10.0	0.77	0.98	0.02	0.38
	IncrementalTopo	7.6	0.89	0.99	0.08	0.66
	TSGM	**1.0**	**0.95**	0.93	0.08	0.60
	PRISM-TopoMap	**1.0**	0.90	0.99	0.13	0.85
Средний	RTAB-Map	**1.0**	0.81	–	–	**0.89**
	ORB-SLAM3	2.2	**0.95**	**1.00**	0.00	0.74
	GLIM	**1.0**	0.80	0.94	**0.27**	0.72
	Hydra	9.6	0.80	0.97	0.01	0.34
	IncrementalTopo	11.4	0.74	0.99	0.08	0.38
	TSGM	**1.0**	**0.95**	0.93	0.08	0.60
	PRISM-TopoMap	**1.0**	0.92	0.99	0.14	0.87
Большой	RTAB-Map	1.6	0.68	–	–	0.61
	ORB-SLAM3	2.2	**0.95**	**1.00**	0.00	0.74
	GLIM	**1.0**	0.80	0.94	**0.27**	0.72
	Hydra	11.8	0.83	0.97	0.01	0.41
	IncrementalTopo	17.2	0.43	0.98	0.07	0.20
	TSGM	**1.0**	**0.95**	0.93	0.08	0.60
	PRISM-TopoMap	**1.0**	0.93	0.99	0.11	**0.92**

личивались, в то время как у базовых алгоритмов – уменьшались. Это связано с тем, что с ростом уровня шума добавлялось больше локаций в граф, что привело к лучшему покрытию сцены и появлению более коротких путей.

Топометрические алгоритмы IncrementalTopo и Hydra построили несвязные графы даже при нулевом уровне шума. Метрический алгоритм ORB-SLAM3 не использовал данную на вход зашумленную позицию, поэтому при всех уровнях шума показал одинаковые результаты, построив несвязные карты на трех сценах из-за сбоя локализации. Метрический алгоритм GLIM тоже не использовал данную на вход зашумленную позицию, поэтому показал одинаковые результаты при всех уровнях шума. Он построил связные карты на всех сценах, но на одной из сцен покрыл только 25% ее площади из-за сбоя картирования, что привело к низким значениям метрик покрытия и SPL. Чисто топологический алгоритм TSGM построил связные графы локаций на всех

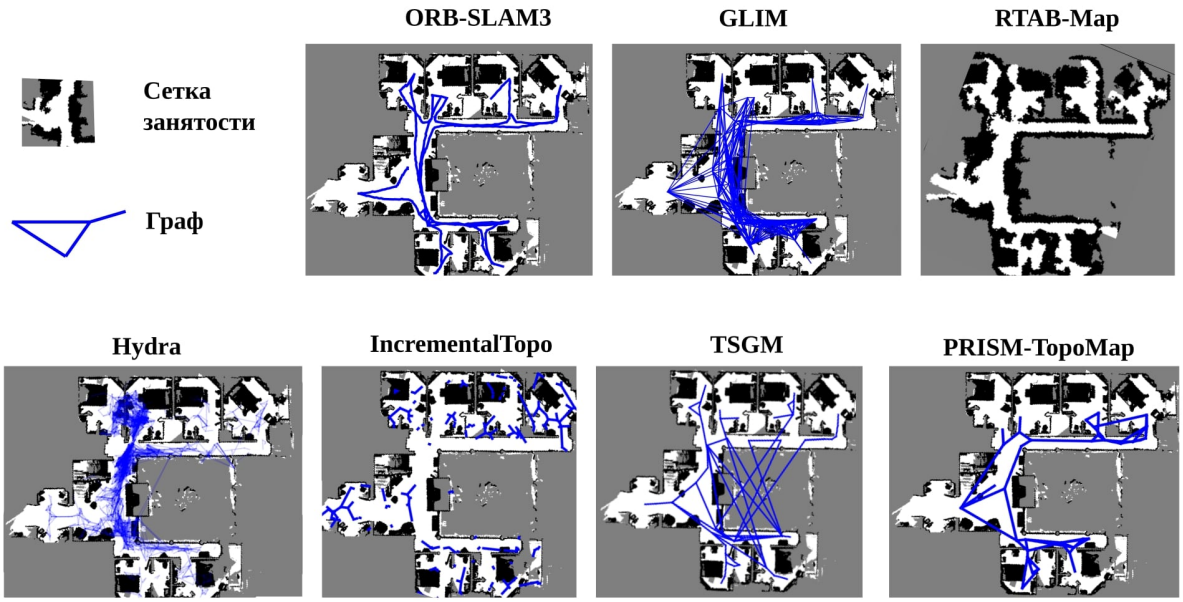

Рисунок 5.6 — Карты, построенные базовыми алгоритмами и алгоритмом PRISM-TopoMap, в сравнении с двумерной сеткой занятости.

сценах с высоким покрытием сцены, но соединил ребрами несмежные локации, что привело к низкому значению SPL (0.60).

На всех уровнях шума алгоритм PRISM-TopoMap показал высокий уровень корректности ребер графа (99%), благодаря использованию оригинального алгоритма сопоставления сканов. Алгоритм ORB-SLAM3 показал значение корректности ребер в 100%, при этом полнота ребер составила менее 1% (граф локаций на нескольких сценах представлял собой цепочку, в которой каждая локация соединялась лишь с предыдущей и следующей локациями в порядке обхода). По полноте ребер графа алгоритм PRISM-TopoMap уступил только алгоритму GLIM, однако алгоритм GLIM показал низкую корректность ребер (94%), что привело к низкому значению путевой эффективности.

Для каждого из рассмотренных алгоритмов было измерено количество оперативной памяти, потребляемой на хранение карты, и постоянной памяти, которую занимает на диске карта, сохраненная после прохождения всего маршрута. Помимо памяти, было измерено среднее время, затраченное на обновление карты по входным данным, и среднее время замыкания цикла (если замыкание циклов было предусмотрено алгоритмом). Все замеры проводились на первой сцене из набора (площадь сцены 345 м^2, длина маршрута 221 м). Для замеров был использован компьютер со следующими характеристиками:

– Процессор: Intel Core i5-9500F, 6 ядер;
– Оперативная память: 32 ГБ, тип DDR4, 2400 МГц;

– Постоянная память: 500 ГБ, твердотельный накопитель (SSD);

– Графический ускоритель: NVidia GeForce GTX 2060, 6 ГБ видеопамяти.

Для определения количества оперативной памяти, занимаемой картой, использовалась разность между количеством оперативной памяти, потребляемой алгоритмом на старте и в конце проезда. При измерении времени работы использовалось реальное время в миллисекундах, прошедшее от получения алгоритмом входных данных до обновления карты.

Результаты измерений представлены в таблице 17. Из таблицы видно, что топологические карты наиболее эффективны по памяти и по времени обновления. Предложенный алгоритм PRISM-TopoMap показал наименьшие затраты памяти на хранение карты среди всех рассмотренных алгоритмов. Граф локаций для помещения площадью более 300 м^2 занял всего 400 кБ. Потребление оперативной памяти на построение карты у PRISM-TopoMap оказалось также невысоким – 150 МБ. При этом граф локаций обновлялся с частотой 9 Гц, что достаточно для работы в реальном времени (при стандартной частоте работы лидара в 10 Гц в обновлении карты участвует почти каждый лидарный скан). Таким образом, PRISM-TopoMap позволяет картировать в реальном времени большие пространства (вплоть до нескольких квадратных километров) и хранить построенные карты на борту робота.

Таблица 17 — Результаты измерений потребления времени и памяти у рассмотренных алгоритмов

Алгоритм	ОЗУ, МБ	Размер карты, МБ	Время обновления, мс	Время замыкания цикла, мс
RTAB-Map	400	7.0	150	300
ORB-SLAM3	600	237	130	250
GLIM	300	3.4	**40**	**40**
Hydra	2200	195	900	2000
IncrementalTopo	40	32	300	-
TSGM	**30**	15	80	80
PRISM-TopoMap	150	**0.4**	110	110

Топометрический алгоритм IncrementalTopo показал низкие затраты памяти на картирование за счет представления карты в виде сетки занятости и диаграммы Вороного, однако время обновления такой карты составило 300 мс. Иерархический топометрический алгоритм Hydra показал наихудшие результаты как по потреблению оперативной памяти, так и по времени, за счет

картирования полной информации о сцене и построения плотной трехмерной модели сцены. Алгоритмы RTAB-Map и GLIM построили компактные метрические карты, однако потребляли в процессе построения этих карт 400 и 300 МБ оперативной памяти соответственно за счет использования большого количества особых точек для оптимизации. Метрическая карта, построенная алгоритмом ORB-SLAM3, заняла более 200 МБ в хранилище за счет сохранения всех дескрипторов особых точек на каждом ключевом кадре.

Таким образом, по результатам экспериментов в симуляционных средах, разработанный алгоритм PRISM-TopoMap оказался способен строить связные топологические карты с высокой долей покрытия сцены и высокой путевой эффективностью, пригодные для планирования путей в реальном времени. Качество работы алгоритма PRISM-TopoMap по всем показателям не ухудшалось с ростом шума позиционирования, в отличие от метрических и топометрических алгоритмов. PRISM-TopoMap оказался наиболее эффективным с точки зрения занимаемого картой объема памяти, что дает возможность применять его для картирования и локализации в средах большой площади.

Эксперименты с навигацией по топологической карте Для оценки эффективности навигации и планирования пути с использованием топологической карты было проведено экспериментальное исследование в симуляционной сцене, представляющей собой модель пятого этажа корпуса Цифра Московского физико-технического института. Сцена состояла из нескольких длинных коридоров и холлов общей площадью около 1600 м^2. В начале осуществлялся проезд по предварительно заданному маршруту длиной 1 км для построения карты в реальном времени. Затем было проведено 20 попыток навигации между случайно заданными точками сцены. Среднее расстояние между начальной и конечной точкой при навигации составляло 78 м. Были проведены измерения потребления оперативной памяти на построение карты. Были измерены среднее время планирования пути от начальной до конечной точки и эффективность навигации, измеренная как SPL [50] (англ. Success weighted by Path Length) – отношение длины оптимального пути между стартовой и целевой точками к длине фактически пройденного агентом пути, умноженное на успешность навигации.

Для сравнения с предложенным топологическим подходом использовался метрический алгоритм картирования и локализации RTAB-Map. Планирование

Таблица 18 — Результаты экспериментов с навигацией в построенной карте

Алгоритм	Потребление памяти, МБ	Время планирования, мс	SPL
RTAB-Map	350	830	0.95
PRISM-TopoMap	**60**	**6**	**0.89**

пути в топологической карте, построенной алгоритмом PRISM-TopoMap, осуществлялось в соответствии с подходом, описанным в разделе 3.1. На нижнем уровне для планирования локального пути использовался алгоритм Theta* [100] на объединенных проекциях сканов текущей локации и соседних с ней локаций. Планирование пути по сетке занятости, построенной алгоритмом RTAB-Map, осуществлялось с помощью алгоритма Theta*. Результаты навигационных экспериментов представлены в таблице 18. Как видно из таблицы, планирование пути в топологической карте происходит более чем в 100 раз быстрее, чем в глобальной метрической карте. При построении топологической карты используется в 6 раз меньше оперативной памяти, чем при построении метрической карты алгоритмом RTAB-Map. При этом по навигационной эффективности алгоритм PRISM-TopoMap слегка уступает метрическому алгоритму RTAB-Map за счет разреженности графа локаций. Навигационная эффективность может быть повышена за счет повышения порога $t_{overlap}$ и, как следствие, большей плотности графа локаций.

5.3 Эксперименты на данных с реальных роботов

Эксперименты на данных с реальных роботов внутри помещений Помимо симуляционных сцен, алгоритм PRISM-TopoMap был апробирован на данных с двух проездов реальных роботов внутри помещений. Первый проезд был осуществлен на роботе Clearpath Husky по пятому этажу корпуса Цифра Московского физико-технического института. Длина траектории составила 212 м. Робот был оснащен 16-лучевым лидаром Velodyne VLP-16, а также камерой Zed 2i, направленной вперед, и камерой Intel Realsense d435, направленной назад. Второй проезд был осуществлен на роботе AgileX Scout Mini в робототехническом центре ФИЦ ИУ РАН. Длина траектории составила 202 м. Робот был

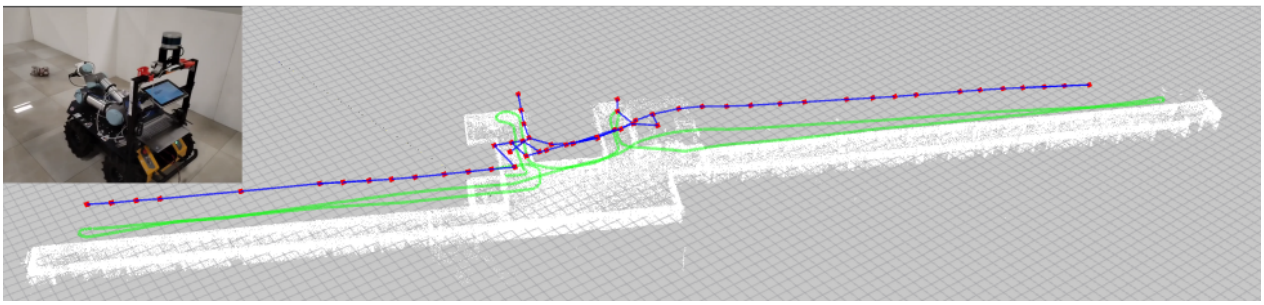

Рисунок 5.7 — Робот Clearpath Husky (слева сверху) и граф локаций, построенный по проезду на нем (красные вершины, синие ребра), в сравнении с метрической картой пройденного помещения (показана белым). Зеленым показана пройденная роботом траектория.

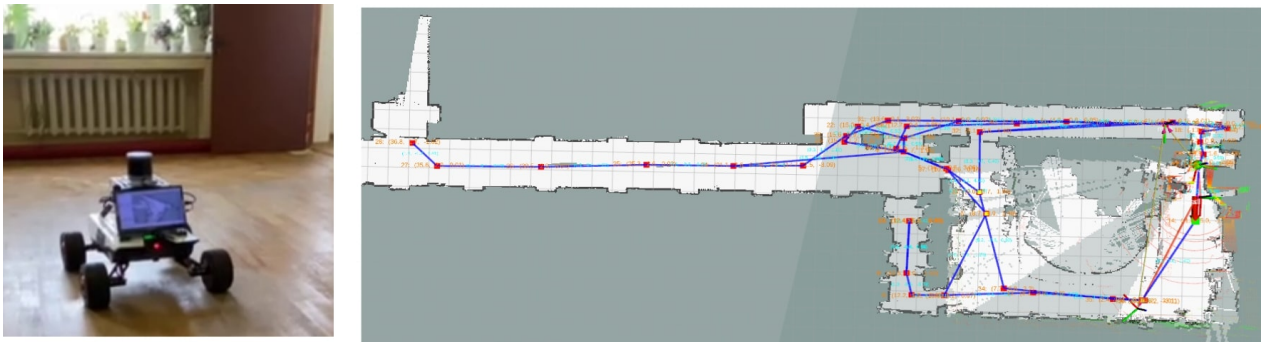

Рисунок 5.8 — Робот AgileX Scout Mini (слева) и граф локаций, построенный по проезду на нем (красные вершины, синие ребра), в сравнении с метрической картой помещения в виде сетки занятости (белым показано свободное пространство, черным – занятое, серым – неизвестное).

оснащен 16-лучевым лидаром Robosense и камерой Intel Realsense d435i, направленной вперед. Управление роботом в обоих проездах осуществлялось вручную.

Данные с обоих проездов подавались на вход алгоритму PRISM-TopoMap в режиме реального времени. В обоих случаях на вход алгоритму подавалась одометрия, рассчитанная по данным о вращении колес. С робота Clearpath Husky на вход подавались облака точек с лидара и изображения с передней и задней камер. С робота AgileX Scout Mini помимо данных одометрии на вход подавались только облака точек с лидара. Карты, построенные по данным с роботов Clearpath Husky и AgileX Scout Mini, представлены на рисунках 5.7 и 5.8 соответственно. В обоих случаях алгоритм PRISM-TopoMap успешно построил граф локаций и замкнул циклы, несмотря на значительную погрешность колесной одометрии и присутствие динамических объектов (движущихся людей) рядом с роботом.

Эксперименты на данных с реального робота в открытых пространствах Для оценки работоспособности алгоритма PRISM-TopoMap при долговременной навигации в средах большой площади были проведены эксперименты на наборе данных ITLP-Campus Outdoor [8]. Набор состоял из 9 проездов описанного выше робота Clearpath Husky по территории кампуса МФТИ в разные времена суток и времена года. Длина каждого из проездов превышала 3 км. Все проезды были сделаны по одному и тому же маршруту с незначительными отклонениями (в пределах 10 м). По первым четырем проездам с помощью алгоритмической пост-обработки были вычислены координаты опорных точек, расположенных по маршруту с интервалом в 5 м, в общей глобальной системе координат.

В ходе экспериментов решалась задача метрической локализации (определения метрических координат робота в каждый момент времени) по предварительно построенной карте. Карта строилась по проезду с индексом 03 (весна, день), вершинами являлись опорные точки маршрута с известными координатами. Для локализации использовались проезды с индексами 00 (зима, сумерки), 01 (зима, день) и 02 (весна, ночь). В качестве источника одометрии использовался алгоритм GLIM [13] без построения карты. Для повышения устойчивости локализации к смене сезона и динамическим объектам в процедуру сопоставления сканов, описанную в разделе 3.2, были добавлены точки бордюров. Алгоритм обнаружения бордюров и процедура сопоставления сканов с учетом бордюров описаны в работе [9].

Для работы в режиме локализации по предварительно построенной карте из описанной в разделе 3.2 процедуры обновления графа алгоритма PRISM-TopoMap был убран пункт 4 (добавление новой локации в граф). Если не удалось сменить локацию v_{cur} по ребру или по результатам локализации, то значение v_{cur} не менялось, а относительное положение T_{cur} менялось по данным одометрии. Для избегания ложных локализаций применялось следующее правило: если последняя успешная смена значения v_{cur} по ребру или по результатам локализации была δ_t секунд назад, и расстояние от текущего положения робота (вычисленного по одометрии) до локализованной вершины графа превосходит $c \cdot \delta_t$, то локализация в этой вершине считается ложной, и вершина удаляется из списка локализованных. Таким образом, текущее состояние робота в графе обновлялось по данным одометрии до момента перехода по ребру или до появления локализации, не противоречащей текущему состоянию.

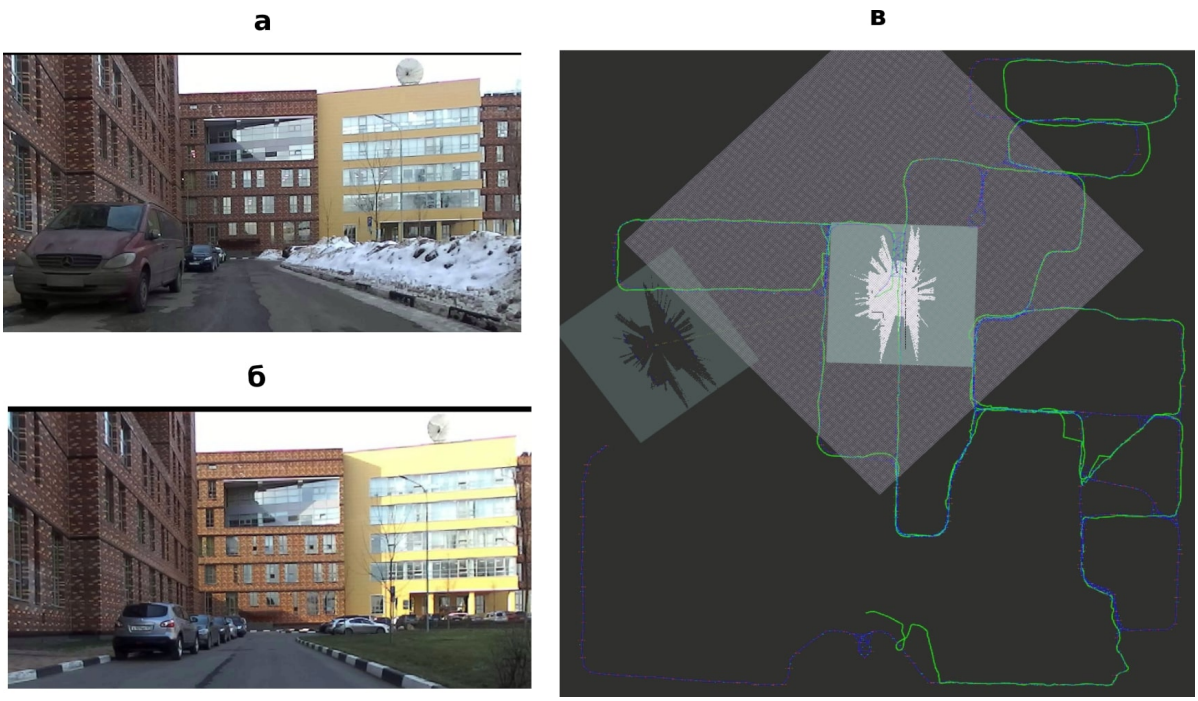

Рисунок 5.9 — (а) Изображение, снятое с робота при проезде 00 (по которому выполняется локализация); (б) Изображение, снятое с робота при проезде 03 (по которому составлена карта); (в) траектория робота на проезде 00, восстановленная с помощью алгоритма PRISM-TopoMap

В ходе эксперимента оценивалась ресурсная эффективность локализации: потребление оперативной памяти при локализации, размер карты в хранилище, среднее время одной локализации. Качество локализации оценивалось по двум критериям: средняя абсолютная ошибка траектории (англ. Mean ATE) и доля успешности локализации SR_{loc}. Локализация считалась успешной, если ошибка не превышала 10 м. Было проведено сравнение качества и ресурсной эффективности локализации с метрическим подходом (локализация с помощью алгоритма HDL [101] в карте, построенной алгоритмом GLIM [13]) и алгоритмом распознавания места BEVPlace++ [77].

Результаты сравнения приведены в таблице 19. Алгоритм HDL, показавший наилучшее потребление оперативной памяти, выдал сбой локализации по проезде 1 км от точки старта – таким образом, успешность его локализации составила всего 30%. Алгоритм BEVPlace++ показал высокую ошибку локализации за счет отсутствия проверки согласованности результатов локализации с текущим состоянием робота. Алгоритм PRISM-TopoMap показал лучшие результаты по всем показателям, кроме потребления оперативной памяти, достигнув средней ошибки локализации 2 м с использованием карты размером

Таблица 19 — Результаты экспериментального исследования локализации в открытых пространствах

Алгоритм	Mean ATE, м	SR_{loc}	Потребление ОЗУ, МБ	Размер карты, МБ	Время, с
GLIM + HDL loc.	3.6*	0.3	**1200****	400	3.0
BEVPlace++	14.4	0.84	15500	15300	1.0
PRISM-TopoMap	**2.0**	**0.98**	4100	**20**	**0.5**

* ATE посчитано на отрезке траектории до момента потери локализации

** Потребление оперативной памяти алгоритмом GLIM при построении карты составило 11 ГБ

всего в 20 МБ на маршрутах длиной 3 км. Восстановленная с помощью PRISM-TopoMap траектория на проезде с индексом 00 показана на рисунке 5.9.

5.4 Выводы по главе

Для оценки качества и эффективности предложенного алгоритма PRISM-TopoMap было проведено обширное экспериментальное исследование в симуляционных помещениях с зашумлением данных одометрии, а также на данных с реальных робототехнических систем внутри и вне помещений. В ходе экспериментов было проведено сравнение алгоритма PRISM-TopoMap с другими метрическими и топологическими алгоритмами. Для проведения численных экспериментов была создана экспериментальная среда в симуляторе Habitat с моделированием ошибки реальной одометрии путем добавления гауссовского шума в соответствии с формулами 2.9, 2.13. Такая модель позволяет оценить устойчивость алгоритмов ОКЛ к накоплению ошибки одометрии, которое присутствует на всех робототехнических системах.

Эксперименты в симуляционных помещениях показали, что PRISM-TopoMap устойчив к шуму одометрии и строит связные графы с высокой долей покрытия сцены в реальном времени. Успешность планирования пути составила 85% при отсутствии шума одометрии и 92% с высокой степенью шума, что оказалось наилучшим результатом среди всех рассмотренных алгоритмов. При этом топологическая карта помещения площадью более 300 м^2, построенная алгоритмом, заняла в хранилище объем в 0.4 МБ – меньше, чем объем одного

исходного облака точек, подаваемого на вход. В ходе экспериментов с навигацией по топологической карте, построенной алгоритмом PRISM-TopoMap, была достигнута навигационная эффективность в 89% по метрике SPL. При этом среднее время планирования пути по карте помещения площадью 1600 м^2 составило всего 6 мс, более чем в 100 раз превзойдя время планирования пути по глобальной метрической карте.

По результатам экспериментов на данных с реальных робототехнических систем, алгоритм PRISM-TopoMap успешно построил топологические карты помещений большой площади в реальном времени, а также выполнил локализацию робота по топологической карте на маршруте длиной 3 км по территории кампуса размером 500x500 м. Успешность локализации составила 98%, среднее значение ошибки локализации – 2.0 м, что значительно превосходит результаты других современных алгоритмов локализации, использованных для сравнения. При этом размер карты, по которой проводилась локализация, составил всего 20 МБ.

Таким образом, разработанный алгоритм PRISM-TopoMap оказался пригодным для построения компактных топологических карт сред большой площади внутри и вне помещений, локализации и навигации по построенным картам. Отсутствие накопления ошибки позиционирования, низкое потребление памяти и вычислительных ресурсов дают возможность использовать PRISM-TopoMap для локализации и навигации робототехнических систем и автономных транспортных средств без использования внешних данных и источников позиционирования.

Заключение

Работа посвящена задаче топологического картирования и локализации в контексте навигации мобильных робототехнических систем. Основные результаты работы заключаются в следующем:

1. Предложена математическая модель задачи топологического картирования и локализации и оценки качества ее решения. В предложенной модели карта представляется в виде графа локаций, а состояние робота в графе – в виде локации и положения робота внутри нее. Качество решения задачи оценивается с помощью путевой эффективности построенного графа, а качество локализации – как точность определения локации, в которой находится робот. Предложенные критерии качества позволяют оценить эффективность применения алгоритма ОКЛ для навигации робота.
2. Разработан алгоритм топологического картирования и локализации. Алгоритм гарантирует связность построенного графа локаций и обеспечивает вычислительно эффективное замыкание циклов за счет процедуры локализации. Отличительной особенностью разработанного алгоритма является фильтрация ложных результатов нейросетевой локализации с помощью оригинальной процедуры сопоставления проекций облаков точек. Разработанный алгоритм обеспечивает низкие затраты памяти, надежную локализацию и высокую путевую эффективность построенной карты.
3. На основе разработанного алгоритма был создан программный комплекс, позволяющий запускать алгоритм на различных робототехнических платформах и симуляторах и оценивать его качество с помощью предложенной математической модели.
4. Выполнено исследование созданного программного комплекса, включающее численные эксперименты в симуляторе и натурные испытания на данных с реальных роботов. В ходе исследования выполнено сравнение с другими современными алгоритмами ОКЛ по предложенным критериям качества и оценка вычислительной эффективности алгоритмов. Путевая эффективность предложенного алгоритма при высокой степени зашумленности одометрии составила 92%, а успешность ло-

кализации на данных с реального робота на траектории длиной 3 км составила 98%, что значительно превосходит результаты других алгоритмов ОКЛ. Высокая вычислительная эффективность разработанного алгоритма подтверждается низким временем обновления карты (110 мс) и низким объемом памяти, занимаемом построенной картой (400 КБ при картировании симуляционных помещений и 20 МБ при картировании открытой среды размером 500х500 м).

Список публикаций автора

1. *Muravyev K.*, *Yakovlev K.* Evaluation of RGB-D SLAM in large indoor environments [Текст] // International Conference on Interactive Collaborative Robotics. — Springer. 2022. — С. 93—104. **Scopus**.
2. *Muravyev K.*, *Yakovlev K.* Evaluation of topological mapping methods in indoor environments [Текст] // IEEE Access. — 2023. — Т. 11. — С. 132683—132698. **Scopus (Q1)**.
3. *Muravyev K.*, *Yakovlev K.* Maintaining topological maps for mobile robots [Текст] // Sixteenth International Conference on Machine Vision (ICMV 2023). Т. 13072. — SPIE. 2024. — С. 265—272. **Scopus**.
4. *Muravyev K.*, *Melekhin A.*, *Yudin D.*, *Yakovlev K.* PRISM-TopoMap: online topological mapping with place recognition and scan matching [Текст] // IEEE Robotics and Automation Letters. — 2025. — С. 3126—3133. **Scopus (Q1)**.
5. *Муравьев К. Ф.* Топологическое картирование помещений с использованием нейросетевой локализации и сопоставления сканов [Текст] // Информационные технологии и вычислительные системы. — 2024. — № 3. — С. 28—38. **ВАК (К1)**.
6. *Muravyev K.*, *Yakovlev K.* NavTopo: Leveraging Topological Maps for Autonomous Navigation of a Mobile Robot [Текст] // International Conference on Interactive Collaborative Robotics. — Springer. 2024. — С. 144—157. **Scopus**.
7. *Муравьев К.*, *Алхаддад М.*, *Панов А.*, *Миронов К.* Иерархическая навигация с избеганием препятствий и прохождением проёмов на четырёхколёсном мобильном роботе [Текст] // XIV Всероссийское совещание по проблемам управления. — 2024. — С. 1640—1644.
8. *Melekhin A.*, *Bezuglyj V.*, *Petryashin I.*, *Muravyev K.*, *Linok S.*, *Yudin D.*, *Panov A.* ITLP-Campus: A Dataset for Multimodal Semantic Place Recognition [Текст] // International Conference on Intelligent Information Technologies for Industry. — Springer. 2024. — С. 185—195. **Scopus**.

9. *Muravyev K.*, *Yuryev V.*, *Bulichev O.*, *Yudin D.*, *Yakovlev K.* PRISM-Loc: a Lightweight Long-range LiDAR Localization in Urban Environments with Topological Maps [Текст]. — 2025. — arXiv: 2506.15849 [cs.RO]. — URL: https://arxiv.org/abs/2506.15849.

Список литературы

10. *Campos C.*, *Elvira R.*, *Rodríguez J. J. G.*, *Montiel J. M.*, *Tardós J. D.* Orb-slam3: An accurate open-source library for visual, visual–inertial, and multimap slam [Текст] // IEEE Transactions on Robotics. — 2021. — Т. 37, № 6. — С. 1874—1890.
11. *Zhang J.*, *Singh S.* [и др.]. LOAM: Lidar odometry and mapping in real-time. [Текст] // Robotics: Science and systems. Т. 2. — Berkeley, CA. 2014. — С. 1—9.
12. *Qin T.*, *Shen S.* Online temporal calibration for monocular visual-inertial systems [Текст] // 2018 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS). — IEEE. 2018. — С. 3662—3669.
13. *Koide K.*, *Yokozuka M.*, *Oishi S.*, *Banno A.* Glim: 3d range-inertial localization and mapping with gpu-accelerated scan matching factors [Текст] // Robotics and Autonomous Systems. — 2024. — Т. 179. — С. 104750.
14. *Thrun S.* [и др.]. Robotic mapping: A survey [Текст]. — 2002.
15. *Elfes A.* Sonar-based real-world mapping and navigation [Текст] // IEEE Journal on Robotics and Automation. — 1987. — Т. 3, № 3. — С. 249—265.
16. *Labbé M.*, *Michaud F.* RTAB-Map as an open-source lidar and visual simultaneous localization and mapping library for large-scale and long-term online operation [Текст] // Journal of Field Robotics. — 2019. — Т. 36, № 2. — С. 416—446.
17. *Hart P. E.*, *Nilsson N. J.*, *Raphael B.* A formal basis for the heuristic determination of minimum cost paths [Текст] // IEEE transactions on Systems Science and Cybernetics. — 1968. — Т. 4, № 2. — С. 100—107.
18. *Roth-Tabak Y.*, *Jain R.* Building an environment model using depth information [Текст] // Computer. — 1989. — Т. 22, № 6. — С. 85—90.
19. *Reich C.*, *Ritter R.*, *Thesing J.* 3-D shape measurement of complex objects by combining photogrammetry and fringe projection [Текст] // Optical Engineering. — 2000. — Т. 39, № 1. — С. 224—231.

20. *Hess W.*, *Kohler D.*, *Rapp H.*, *Andor D.* Real-time loop closure in 2D LIDAR SLAM [Текст] // 2016 IEEE international conference on robotics and automation (ICRA). — IEEE. 2016. — С. 1271—1278.

21. *Oleynikova H.*, *Taylor Z.*, *Fehr M.*, *Siegwart R.*, *Nieto J.* Voxblox: Incremental 3d euclidean signed distance fields for on-board mav planning [Текст] // 2017 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS). — IEEE. 2017. — С. 1366—1373.

22. *Kuipers B.*, *Byun Y.-T.* A robot exploration and mapping strategy based on a semantic hierarchy of spatial representations [Текст] // Robotics and autonomous systems. — 1991. — Т. 8, № 1/2. — С. 47—63.

23. *Mataric M. J.* A distributed model for mobile robot environment-learning and navigation [Текст]. — 1990.

24. *Shatkay H.*, *Kaelbling L. P.* Learning topological maps with weak local odometric information [Текст] // IJCAI (2). — Citeseer. 1997. — С. 920—929.

25. *Blochliger F.*, *Fehr M.*, *Dymczyk M.*, *Schneider T.*, *Siegwart R.* Topomap: Topological mapping and navigation based on visual slam maps [Текст] // 2018 IEEE International Conference on Robotics and Automation (ICRA). — IEEE. 2018. — С. 3818—3825.

26. *Niijima S.*, *Umeyama R.*, *Sasaki Y.*, *Mizoguchi H.* City-scale grid-topological hybrid maps for autonomous mobile robot navigation in urban area [Текст] // 2020 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS). — IEEE. 2020. — С. 2065—2071.

27. *He Z.*, *Sun H.*, *Hou J.*, *Ha Y.*, *Schwertfeger S.* Hierarchical topometric representation of 3D robotic maps [Текст] // Autonomous Robots. — 2021. — Т. 45, № 5. — С. 755—771.

28. *Mielle M.*, *Magnusson M.*, *Lilienthal A. J.* A method to segment maps from different modalities using free space layout maoris: map of ripples segmentation [Текст] // 2018 IEEE International Conference on Robotics and Automation (ICRA). — IEEE. 2018. — С. 4993—4999.

29. *Liao Z.*, *Zhang Y.*, *Luo J.*, *Yuan W.* TSM: Topological scene map for representation in indoor environment understanding [Текст] // IEEE Access. — 2020. — Т. 8. — С. 185870—185884.

30. *Chen X.*, *Zhou B.*, *Lin J.*, *Zhang Y.*, *Zhang F.*, *Shen S.* Fast 3D sparse topological skeleton graph generation for mobile robot global planning [Текст] // 2022 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS). — IEEE. 2022. — С. 10283—10289.

31. *Wiyatno R. R.*, *Xu A.*, *Paull L.* Lifelong topological visual navigation [Текст] // IEEE Robotics and Automation Letters. — 2022. — Т. 7, № 4. — С. 9271—9278.

32. *Schmid L.*, *Reijgwart V.*, *Ott L.*, *Nieto J.*, *Siegwart R.*, *Cadena C.* A unified approach for autonomous volumetric exploration of large scale environments under severe odometry drift [Текст] // IEEE Robotics and Automation Letters. — 2021. — Т. 6, № 3. — С. 4504—4511.

33. *Reijgwart V.*, *Millane A.*, *Oleynikova H.*, *Siegwart R.*, *Cadena C.*, *Nieto J.* Voxgraph: Globally consistent, volumetric mapping using signed distance function submaps [Текст] // IEEE Robotics and Automation Letters. — 2019. — Т. 5, № 1. — С. 227—234.

34. *Hughes N.*, *Chang Y.*, *Carlone L.* Hydra: a real-time spatial perception system for 3d scene graph construction and optimization [Текст]. — 2022.

35. *Rosinol A.*, *Gupta A.*, *Abate M.*, *Shi J.*, *Carlone L.* 3D dynamic scene graphs: Actionable spatial perception with places, objects, and humans [Текст] // arXiv preprint arXiv:2002.06289. — 2020.

36. *Rosinol A.*, *Violette A.*, *Abate M.*, *Hughes N.*, *Chang Y.*, *Shi J.*, *Gupta A.*, *Carlone L.* Kimera: From SLAM to spatial perception with 3D dynamic scene graphs [Текст] // The International Journal of Robotics Research. — 2021. — Т. 40, № 12—14. — С. 1510—1546.

37. *Oleynikova H.*, *Taylor Z.*, *Siegwart R.*, *Nieto J.* Sparse 3d topological graphs for micro-aerial vehicle planning. In 2018 IEEE [Текст] // RSJ International Conference on Intelligent Robots and Systems (IROS). — С. 1—9.

38. *Gálvez-López D.*, *Tardos J. D.* Bags of binary words for fast place recognition in image sequences [Текст] // IEEE Transactions on robotics. — 2012. — Т. 28, № 5. — С. 1188—1197.

39. *Fischler M. A.*, *Bolles R. C.* Random sample consensus: a paradigm for model fitting with applications to image analysis and automated cartography [Текст] // Communications of the ACM. — 1981. — Т. 24, № 6. — С. 381—395.

40. *Yang H.*, *Shi J.*, *Carlone L.* Teaser: Fast and certifiable point cloud registration [Текст] // IEEE Transactions on Robotics. — 2020. — Т. 37, № 2. — С. 314—333.

41. *Schmid L.*, *Abate M.*, *Chang Y.*, *Carlone L.* Khronos: A unified approach for spatio-temporal metric-semantic slam in dynamic environments [Текст] // arXiv preprint arXiv:2402.13817. — 2024.

42. *Bavle H.*, *Sanchez-Lopez J. L.*, *Shaheer M.*, *Civera J.*, *Voos H.* S-graphs+: Real-time localization and mapping leveraging hierarchical representations [Текст] // IEEE Robotics and Automation Letters. — 2023. — Т. 8, № 8. — С. 4927—4934.

43. *Chaplot D. S.*, *Salakhutdinov R.*, *Gupta A.*, *Gupta S.* Neural topological slam for visual navigation [Текст] // Proceedings of the IEEE/CVF conference on computer vision and pattern recognition. — 2020. — С. 12875—12884.

44. *Savinov N.*, *Dosovitskiy A.*, *Koltun V.* Semi-parametric topological memory for navigation [Текст] // arXiv preprint arXiv:1803.00653. — 2018.

45. *Kwon O.*, *Kim N.*, *Choi Y.*, *Yoo H.*, *Park J.*, *Oh S.* Visual graph memory with unsupervised representation for visual navigation [Текст] // Proceedings of the IEEE/CVF international conference on computer vision. — 2021. — С. 15890—15899.

46. *Kim N.*, *Kwon O.*, *Yoo H.*, *Choi Y.*, *Park J.*, *Oh S.* Topological semantic graph memory for image-goal navigation [Текст] // Conference on Robot Learning. — PMLR. 2023. — С. 393—402.

47. *Chen K.*, *Chen J. K.*, *Chuang J.*, *Vázquez M.*, *Savarese S.* Topological planning with transformers for vision-and-language navigation [Текст] // Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition. — 2021. — С. 11276—11286.

48. *An D.*, *Wang H.*, *Wang W.*, *Wang Z.*, *Huang Y.*, *He K.*, *Wang L.* Etpnav: Evolving topological planning for vision-language navigation in continuous environments [Текст] // IEEE Transactions on Pattern Analysis and Machine Intelligence. — 2024.

49. *Xia F.*, *Zamir A. R.*, *He Z.*, *Sax A.*, *Malik J.*, *Savarese S.* Gibson env: Real-world perception for embodied agents [Текст] // Proceedings of the IEEE conference on computer vision and pattern recognition. — 2018. — С. 9068—9079.

50. *Anderson P.*, *Chang A.*, *Chaplot D. S.*, *Dosovitskiy A.*, *Gupta S.*, *Koltun V.*, *Kosecka J.*, *Malik J.*, *Mottaghi R.*, *Savva M.* [и др.]. On evaluation of embodied navigation agents [Текст] // arXiv preprint arXiv:1807.06757. — 2018.

51. *Gomez C.*, *Fehr M.*, *Millane A.*, *Hernandez A. C.*, *Nieto J.*, *Barber R.*, *Siegwart R.* Hybrid topological and 3d dense mapping through autonomous exploration for large indoor environments [Текст] // 2020 IEEE International Conference on Robotics and Automation (ICRA). — IEEE. 2020. — С. 9673—9679.

52. *Tang L.*, *Wang Y.*, *Ding X.*, *Yin H.*, *Xiong R.*, *Huang S.* Topological local-metric framework for mobile robots navigation: a long term perspective [Текст] // Autonomous Robots. — 2019. — Т. 43. — С. 197—211.

53. *Fox D.*, *Burgard W.*, *Dellaert F.*, *Thrun S.* Monte carlo localization: Efficient position estimation for mobile robots [Текст] // Aaai/iaai. — 1999. — Т. 1999, № 343—349. — С. 2—2.

54. *Caselitz T.*, *Steder B.*, *Ruhnke M.*, *Burgard W.* Monocular camera localization in 3d lidar maps [Текст] // 2016 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS). — IEEE. 2016. — С. 1926—1931.

55. *Wang Z.*, *Fang J.*, *Dai X.*, *Zhang H.*, *Vlacic L.* Intelligent vehicle self-localization based on double-layer features and multilayer LIDAR [Текст] // IEEE Transactions on Intelligent Vehicles. — 2020. — Т. 5, № 4. — С. 616—625.

56. *Li L.*, *Yang M.*, *Weng L.*, *Wang C.* Robust localization for intelligent vehicles based on pole-like features using the point cloud [Текст] // IEEE Transactions on Automation Science and Engineering. — 2021. — Т. 19, № 2. — С. 1095—1108.

57. *Cummins M.*, *Newman P.* FAB-MAP: Probabilistic localization and mapping in the space of appearance [Текст] // The International journal of robotics research. — 2008. — Т. 27, № 6. — С. 647—665.

58. *Bay H.*, *Tuytelaars T.*, *Van Gool L.* Surf: Speeded up robust features [Текст] // Computer Vision–ECCV 2006: 9th European Conference on Computer Vision, Graz, Austria, May 7-13, 2006. Proceedings, Part I 9. — Springer. 2006. — С. 404—417.

59. *Arandjelovic R.*, *Gronat P.*, *Torii A.*, *Pajdla T.*, *Sivic J.* NetVLAD: CNN architecture for weakly supervised place recognition [Текст] // Proceedings of the IEEE conference on computer vision and pattern recognition. — 2016. — С. 5297—5307.

60. *Torii A.*, *Sivic J.*, *Pajdla T.*, *Okutomi M.* Visual place recognition with repetitive structures [Текст] // Proceedings of the IEEE conference on computer vision and pattern recognition. — 2013. — С. 883—890.

61. *Berton G.*, *Masone C.*, *Caputo B.* Rethinking visual geo-localization for large-scale applications [Текст] // Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition. — 2022. — С. 4878—4888.

62. *Ali-Bey A.*, *Chaib-Draa B.*, *Giguere P.* Mixvpr: Feature mixing for visual place recognition [Текст] // Proceedings of the IEEE/CVF winter conference on applications of computer vision. — 2023. — С. 2998—3007.

63. *Uy M. A.*, *Lee G. H.* PointNetVLAD: Deep Point Cloud Based Retrieval for Large-Scale Place Recognition [Текст] // CVPR. — 2018. — С. 4470—4479. — (Дата обр. 25.04.2022).

64. *Qi C. R.*, *Su H.*, *Mo K.*, *Guibas L. J.* Pointnet: Deep learning on point sets for 3d classification and segmentation [Текст] // Proceedings of the IEEE conference on computer vision and pattern recognition. — 2017. — С. 652—660.

65. *Komorowski J.* MinkLoc3D: Point Cloud Based Large-Scale Place Recognition [Текст] // Proceedings of the IEEE/CVF Winter Conference on Applications of Computer Vision. — 2021. — P. 1790—1799. — (Visited on 04/06/2023).

66. *Fan Z.*, *Song Z.*, *Liu H.*, *Lu Z.*, *He J.*, *Du X.* SVT-Net: Super Light-Weight Sparse Voxel Transformer for Large Scale Place Recognition [Текст] // AAAI. — 2022. — June. — Vol. 36, no. 1. — P. 551—560. — (Visited on 04/03/2023).

67. *Sattler T.*, *Maddern W.*, *Toft C.*, *Torii A.*, *Hammarstrand L.*, *Stenborg E.*, *Safari D.*, *Okutomi M.*, *Pollefeys M.*, *Sivic J.* [и др.]. Benchmarking 6dof outdoor visual localization in changing conditions [Текст] // Proceedings of the IEEE conference on computer vision and pattern recognition. — 2018. — С. 8601—8610.

68. *Xie S.*, *Pan C.*, *Peng Y.*, *Liu K.*, *Ying S.* Large-Scale Place Recognition Based on Camera-LiDAR Fused Descriptor [Текст] // Sensors. — 2020. — Jan. — Vol. 20, no. 10. — P. 2870. — (Visited on 05/04/2022).

69. *He K.*, *Zhang X.*, *Ren S.*, *Sun J.* Deep residual learning for image recognition [Текст] // Proceedings of the IEEE conference on computer vision and pattern recognition. — 2016. — С. 770—778.

70. *Geiger A.*, *Lenz P.*, *Stiller C.*, *Urtasun R.* Vision meets robotics: The kitti dataset [Текст] // The international journal of robotics research. — 2013. — Т. 32, № 11. — С. 1231—1237.

71. *Komorowski J.*, *Wysoczańska M.*, *Trzcinski T.* MinkLoc++: Lidar and Monocular Image Fusion for Place Recognition [Текст] // 2021 International Joint Conference on Neural Networks (IJCNN). — 07.2021. — С. 1—8.

72. *Melekhin A.*, *Yudin D.*, *Petryashin I.*, *Bezuglyj V.* Mssplace: multi-sensor place recognition with visual and text semantics [Текст] // arXiv preprint arXiv:2407.15663. — 2024.

73. *Besl P. J.*, *McKay N. D.* Method for registration of 3-D shapes [Текст] // Sensor fusion IV: control paradigms and data structures. Т. 1611. — Spie. 1992. — С. 586—606.

74. *Rusu R. B.*, *Blodow N.*, *Beetz M.* Fast point feature histograms (FPFH) for 3D registration [Текст] // 2009 IEEE international conference on robotics and automation. — IEEE. 2009. — С. 3212—3217.

75. *Choy C.*, *Park J.*, *Koltun V.* Fully convolutional geometric features [Текст] // Proceedings of the IEEE/CVF international conference on computer vision. — 2019. — С. 8958—8966.

76. *Luo L.*, *Cao S.-Y.*, *Han B.*, *Shen H.-L.*, *Li J.* Bvmatch: Lidar-based place recognition using bird's-eye view images [Текст] // IEEE Robotics and Automation Letters. — 2021. — Т. 6, № 3. — С. 6076—6083.

77. *Luo L.*, *Cao S.-Y.*, *Li X.*, *Xu J.*, *Ai R.*, *Yu Z.*, *Chen X.* BEVPlace++: Fast, Robust, and Lightweight LiDAR Global Localization for Unmanned Ground Vehicles [Текст] // arXiv preprint arXiv:2408.01841. — 2024.

78. *Carlevaris-Bianco N.*, *Ushani A. K.*, *Eustice R. M.* University of Michigan North Campus long-term vision and lidar dataset [Текст] // The International Journal of Robotics Research. — 2016. — Т. 35, № 9. — С. 1023—1035.

79. *Pramatarov G.*, *De Martini D.*, *Gadd M.*, *Newman P.* BoxGraph: Semantic place recognition and pose estimation from 3D LiDAR [Текст] // 2022 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS). — IEEE. 2022. — С. 7004—7011.

80. *Wang X.*, *Marcotte R. J.*, *Olson E.* GLFP: Global localization from a floor plan [Текст] // 2019 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS). — IEEE. 2019. — С. 1627—1632.

81. *Schmid L.*, *Delmerico J.*, *Schönberger J. L.*, *Nieto J.*, *Pollefeys M.*, *Siegwart R.*, *Cadena C.* Panoptic multi-tsdfs: a flexible representation for online multi-resolution volumetric mapping and long-term dynamic scene consistency [Текст] // 2022 International Conference on Robotics and Automation (ICRA). — IEEE. 2022. — С. 8018—8024.

82. *Боковой А.* Исследование методов и разработка алгоритмов одновременного картирования и локализации по видеопотоку единственной камеры [Текст]. — 2022.

83. *Dijkstra E. W.* [и др.]. A note on two problems in connexion with graphs [Текст] // Numerische mathematik. — 1959. — Т. 1, № 1. — С. 269—271.

84. *Holkar K.*, *Waghmare L. M.* An overview of model predictive control [Текст] // International Journal of control and automation. — 2010. — Т. 3, № 4. — С. 47—63.

85. *Radenović F.*, *Tolias G.*, *Chum O.* Fine-tuning CNN image retrieval with no human annotation [Текст] // IEEE transactions on pattern analysis and machine intelligence. — 2018. — Т. 41, № 7. — С. 1655—1668.

86. *Rublee E.*, *Rabaud V.*, *Konolige K.*, *Bradski G.* ORB: An efficient alternative to SIFT or SURF [Текст] // 2011 International conference on computer vision. — Ieee. 2011. — С. 2564—2571.

87. *Muja M.*, *Lowe D.* Flann-fast library for approximate nearest neighbors user manual [Текст] // Computer Science Department, University of British Columbia, Vancouver, BC, Canada. — 2009. — Т. 5. — С. 6.

88. *Harris C.*, *Stephens M.* [и др.]. A combined corner and edge detector [Текст] // Alvey vision conference. Т. 15. — Citeseer. 1988. — С. 10—5244.

89. *Savva M.*, *Kadian A.*, *Maksymets O.*, *Zhao Y.*, *Wijmans E.*, *Jain B.*, *Straub J.*, *Liu J.*, *Koltun V.*, *Malik J.* [и др.]. Habitat: A platform for embodied ai research [Текст] // Proceedings of the IEEE/CVF international conference on computer vision. — 2019. — С. 9339—9347.

90. *Burri M.*, *Nikolic J.*, *Gohl P.*, *Schneider T.*, *Rehder J.*, *Omari S.*, *Achtelik M. W.*, *Siegwart R.* The EuRoC micro aerial vehicle datasets [Текст] // The International Journal of Robotics Research. — 2016. — Т. 35, № 10. — С. 1157—1163.

91. *Sturm J.*, *Burgard W.*, *Cremers D.* Evaluating egomotion and structure-from-motion approaches using the TUM RGB-D benchmark [Текст] // Proc. of the Workshop on Color-Depth Camera Fusion in Robotics at the IEEE/RJS International Conference on Intelligent Robot Systems (IROS). Т. 13. — 2012. — С. 6.

92. *Koenig N.*, *Howard A.* Design and use paradigms for gazebo, an open-source multi-robot simulator [Текст] // 2004 IEEE/RSJ international conference on intelligent robots and systems (IROS)(IEEE Cat. No. 04CH37566). Т. 3. — Ieee. 2004. — С. 2149—2154.

93. *Makoviychuk V.*, *Wawrzyniak L.*, *Guo Y.*, *Lu M.*, *Storey K.*, *Macklin M.*, *Hoeller D.*, *Rudin N.*, *Allshire A.*, *Handa A.* [и др.]. Isaac gym: High performance gpu-based physics simulation for robot learning [Текст] // arXiv preprint arXiv:2108.10470. — 2021.

94. *Lozano-Pérez T.*, *Wesley M. A.* An algorithm for planning collision-free paths among polyhedral obstacles [Текст] // Communications of the ACM. — 1979. — Т. 22, № 10. — С. 560—570.

95. *Kasmynin K.*, *Mironov K.* Vectorized Visibility Graph Planning with Neural Polygon Extraction [Текст] // International Conference on Interactive Collaborative Robotics. — Springer. 2024. — С. 265—280.

96. *Chang A.*, *Dai A.*, *Funkhouser T.*, *Halber M.*, *Niessner M.*, *Savva M.*, *Song S.*, *Zeng A.*, *Zhang Y.* Matterport3d: Learning from rgb-d data in indoor environments [Текст] // arXiv preprint arXiv:1709.06158. — 2017.

97. *Qin Z.*, *Yu H.*, *Wang C.*, *Guo Y.*, *Peng Y.*, *Ilic S.*, *Hu D.*, *Xu K.* Geotransformer: Fast and robust point cloud registration with geometric transformer [Текст] // IEEE Transactions on Pattern Analysis and Machine Intelligence. — 2023.

98. *Yuan Y.*, *Schwertfeger S.* Incrementally building topology graphs via distance maps [Текст] // 2019 IEEE International Conference on Real-time Computing and Robotics (RCAR). — IEEE. 2019. — С. 468—474.

99. *Xie E.*, *Wang W.*, *Yu Z.*, *Anandkumar A.*, *Alvarez J. M.*, *Luo P.* SegFormer: Simple and efficient design for semantic segmentation with transformers [Текст] // NeurIPS. — 2021. — Т. 34. — С. 12077—12090.

100. *Daniel K.*, *Nash A.*, *Koenig S.*, *Felner A.* Theta*: Any-angle path planning on grids [Текст] // Journal of Artificial Intelligence Research. — 2010. — Т. 39. — С. 533—579.

101. *Koide K.*, *Miura J.*, *Menegatti E.* A portable three-dimensional LIDAR-based system for long-term and wide-area people behavior measurement [Текст] // International Journal of Advanced Robotic Systems. — 2019. — Т. 16, № 2. — С. 1729881419841532.

Приложение А. Свидетельство о госурадственной регистрации программы для ЭВМ № 2025662382

РОССИЙСКАЯ ФЕДЕРАЦИЯ

СВИДЕТЕЛЬСТВО

о государственной регистрации программы для ЭВМ

№ 2025662382

PRISM-ТороМар: программная библиотека топологического картирования с помощью распознавания мест и сопоставления сканов

Правообладатель: ***Федеральное государственное учреждение «Федеральный исследовательский центр «Информатика и управление» Российской академии наук» (RU)***

Автор(ы): ***Муравьев Кирилл Федорович (RU)***

Заявка № **2025660983**
Дата поступления **05 мая 2025 г.**
Дата государственной регистрации
в Реестре программ для ЭВМ ***20 мая 2025 г.***

Руководитель Федеральной службы по интеллектуальной собственности

ДОКУМЕНТ ПОДПИСАН ЭЛЕКТРОННОЙ ПОДПИСЬЮ
Сертификат 0692e7c1a6300bf54f240f670bca2026
Владелец Зубов Юрий Сергеевич
Действителен с 10.07.2024 по 03.10.2025

Ю.С. Зубов

Рисунок А.1 — Свидетельство о регистрации программы для ЭВМ