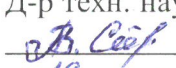


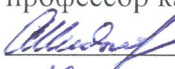
Министерство науки и высшего образования Российской Федерации
НАЦИОНАЛЬНЫЙ ИССЛЕДОВАТЕЛЬСКИЙ ТОМСКИЙ
ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ (НИ ТГУ)
Факультет инновационных технологий (ФИТ)
Кафедра информационного обеспечения инновационной деятельности (ИОИД)

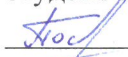
ДОПУСТИТЬ К ЗАЩИТЕ В ГЭК
Руководитель ООП, зав. каф. УК,
Д-р техн. наук, профессор
 В.И. Сырямкин
« 19 » июня 2019 г.

МАГИСТЕРСКАЯ ДИССЕРТАЦИЯ
АЛГОРИТМЫ SLAM-НАВИГАЦИИ В ЗАДАЧАХ СИНТЕЗА АВТОНОМНЫХ
МОБИЛЬНЫХ РОБОТОВ

по основной образовательной программе подготовки магистров
направление подготовки 09.04.02 – Информационные системы и технологии

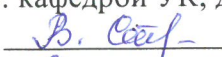
Пославский Сергей

Научный руководитель
Д-р техн. наук,
профессор каф. УК,
 С.В. Шидловский
« 19 » июня 2019 г.

Автор работы
студент группы №18705
 С. Пославский
« 19 » июня 2019 г.

Томск - 2019

МИНИСТЕРСТВО ОБРАЗОВАНИЯ И НАУКИ РФ
НАЦИОНАЛЬНЫЙ ИССЛЕДОВАТЕЛЬСКИЙ ТОМСКИЙ ГОСУДАРСТВЕННЫЙ
УНИВЕРСИТЕТ (НИ ТГУ)
Факультет инновационных технологий (ФИТ)
Кафедра информационного обеспечения инновационной деятельности (ИОИД)

УТВЕРЖДАЮ
Руководитель ООП,
зав. кафедрой УК, д. т. н. профессор
 В.И. Сырянкин
« 9 » апреля 2019 г.

ЗАДАНИЕ

по подготовке выпускной квалификационной работы магистра
студенту группы №18705 Пославскому Сергею

1. Тема ВКР: «Алгоритмы SLAM-навигации в задачах синтеза автономных мобильных роботов» (утверждена приказом по факультету от «09» апреля 2019 г. №30410)

2. Срок сдачи студентом законченной ВКР:

а) на кафедре 17.06.2019 г.

б) в ГЭК 21.06.2019 г.

3. Исходные данные к ВКР: технические средства, используемые в реализации поставленных задач, результаты научно-исследовательской работы, производственных и преддипломной практик.

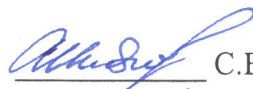
4. Краткое содержание работы: Изучение основ работы с различными алгоритмами SLAM-навигации комплексов пространственной локализации и картографирования. Рассмотрение теоретических аспектов популярных алгоритмов SLAM-навигации. Анализ полученных данных, выбор оптимального алгоритма для текущей задачи. Практическая реализация системы пространственной SLAM-навигации с использованием синтезированной модели мобильного робота, заключение о проделанной работе.

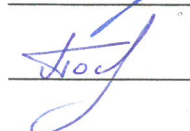
6. Перечень графического материала: результаты исследований и испытаний, представленные в графическом виде, структурная блок-схема подсистемы навигации, структурная схема АСР.

7. Дата выдачи задания 9 апреля 2019 г

Руководитель ВКР,
профессор кафедры УК

Задание принял к исполнению

 С.В. Шидловский

 С. Пославский

РЕФЕРАТ

Диссертация на соискание ученой степени магистра инноватики: 107 с., 37 рис., 2 таб., 58 источников.

SLAM-НАВИГАЦИЯ, VSLAM, ОДОМЕТРИЯ, АЛГОРИТМЫ ПРОСТРАНСТВЕННОЙ НАВИГАЦИИ, ЛОКАЛИЗАЦИЯ И КАРТОГРАФИРОВАНИЕ, LIDAR, АВТОНОМНОЕ УПРАВЛЕНИЕ

В магистерской диссертации рассмотрены основные принципы и особенности разработки мобильных робототехнических систем с использованием алгоритмов SLAM-навигации.

Исследованы различные существующие SLAM-методы комплекса пространственной навигации и позиционирования. Реализованы и исследованы алгоритмы SLAM-навигации, основанные на датчиках визуальной информации и датчиков типа лидар, а также алгоритм обхода препятствий Obstacle Avoidance.

Разработана САУ, целью которой является автоматизация сбора данных, поступаемых с датчиков входной информации, обработка полученных данных, а также автоматизация принятия навигационных решений, основанных на полученных и обработанных данных о состоянии окружающей среды. В состав АСР входят подсистемы сбора и обработки данных, картографирования и локализации, телеуправления, обхода препятствий, Google Cartographer, а также алгоритмы, оптимизирующие работу системы с помощью сегментации данных на получаемой карте окружающей местности.

Для разработки программной части использовалась мета-операционная система ROS, обладающая большим набором инструментов для разработки, проектирования и моделирования различных робототехнических систем.

Проанализированы результаты работы алгоритмов. Выбран оптимальный алгоритм SLAM-навигации для подобного рода задач, а также

синтезирован макет мобильного робота на основе проведенных исследований.

The abstract

Dissertation for the degree of Master of Innovation 107 pages, 37 pictures, 2 tab., 58 sources.

SLAM-NAVIGATION, VSLAM, ODOMETRY, ALGORITHMS OF SPATIAL NAVIGATION, LOCALIZATION AND MAPPING, LIDAR, AUTONOMOUS CONTROL

This Master's Thesis discusses the basic principles and the development of mobile robotic systems using SLAM-navigation algorithms.

Various existing SLAM-methods of spatial navigation and positioning complex have been discussed and investigated. In regards to this, multiple SLAM-navigation algorithms based on visual information sensors and LIDAR-type sensors, as well as the Obstacle Avoidance algorithm have been implemented.

The ACS has been developed, with the purpose to automate the collection of data from input sensors, process the received data, and automate navigation decisions based on the received and processed data on the state of the environment. The ASR includes subsystems for data collection and processing, mapping and localization, teleoperation control, obstacle avoidance, Google Cartographer, as well as algorithms that optimize the system by using data segmentation on the resulting map of the surrounding area.

For the development of the software part, the meta-operating system ROS was used, with a large set of tools for the development, design and simulation of various robotic systems.

After analysing the results of the algorithms, the optimal SLAM-navigation algorithm was chosen, and the layout of robot was designed based on the research performed.

ОГЛАВЛЕНИЕ

ВВЕДЕНИЕ	9
1 Анализ литературных источников	11
1.1 Алгоритмы VSLAM	11
1.2 Элементы VSLAM	13
1.3 Основные модули.....	13
1.4 Дополнительные модули для увеличения стабильности и точности VSLAM.....	14
1.5 Связанные технологии.....	15
1.5.1 Визуальная одометрия.....	16
1.5.2 Структура от движения	16
2 Анализ алгоритмов SLAM-навигации	18
2.1 Функциональные методы	18
2.1.1 MonoSLAM.....	18
2.1.2 PTAM.....	19
2.1.3 Сравнение MonoSLAM и PTAM	21
2.1.4 Методы оптимизации глобальной карты	21
2.2 Прямые методы	22
2.2.1 DTAM.....	22
2.2.2 LSD-SLAM.....	23
2.2.3 SVO и DSO.....	25
2.3 RGB-D vSLAM	25
2.3.1. Разница с монокулярным vSLAM.....	26
2.3.2 KinectFusion	26
2.3.3 SLAM ++	27
2.3.4 Методы RGB-DVO и глобальной оптимизации карт.....	27
2.4 Сравнение и анализ.....	28
3 Современные подходы к решению задачи SLAM для лазерных дальномеров. LOAM	33

3.1 Базовые отличия современного метода LOAM	35
4 Методы и проблемы локализации в задачах синтеза мобильных роботов..	38
4.1 Метод функционального извлечения особых признаков	41
4.1.1 Основная концепция извлечения признаков.....	41
4.1.2 Новый метод поиска кластеров	42
4.1.3 Метод разделение и слияния для извлечения углов и линий.....	44
4.2 Метод WP-ICP	45
4.2.1 Взвешенный параллельный алгоритм оценки положения ICP ...	45
5 Алгоритмы программного комплекса пространственной навигации и мониторинга на основе визуальной одометрии	49
5.1 Описание работы системы визуальной одометрии	49
6 Описание концепции и инструментов фреймворка ROS.....	52
6.1 Общие сведения	52
6.2 Архитектура ROS	53
6.3 Файловая система ROS.....	53
6.4 Уровень вычислительного графа.....	55
7 Разработка автоматизированной системы управления и ее функции	57
7.1 Назначение системы	57
7.2 Описание системы.....	57
7.3 Описание подсистем	58
8 Синтез модели мобильного робота на основе алгоритмов SLAM-навигации .	59
8.1 Выбор структуры автоматической системы регулирования	59
8.2 Разработка структурной схемы подсистемы навигации	61
8.3 Разработка подсистемы телеуправления	65
8.4 Сравнение работы методов VSLAM по качеству визуальной одометрии.....	67
8.5 Реализация и сравнение работы алгоритмов SLAM-навигации основанные на использовании лазерных дальнометров	73
8.6 Интеграция фреймворка Google Cartographer с алгоритмами Obstacle Avoidance.....	81

9 Реализация синтезированной модели	92
ЗАКЛЮЧЕНИЕ	100
СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ	102

ВВЕДЕНИЕ

Мобильные навигационные системы, основанные на современных алгоритмах SLAM-навигации, сегодня широко применяются в разных отраслях. Они решают оборонные задачи, выполняют фотосъемку, картографирование локальных участков местности, а также широко применяются в системах типа “умный дом”. Алгоритмы SLAM-навигации напрямую связаны с СТЗ.

Данное понятие включает в себя область инженерных разработок, направленных на преобразование получаемой входной информации о состоянии окружающей среды и дальнейшее применение результатов полученного преобразования. В базовом представлении, СТЗ представляют собой систему конвертации данных, получаемых с устройств входной визуальной информации, с дальнейшим выполнением различных программно-аппаратных операций над полученной информацией.

Для синтеза высокоинтеллектуального робота, робототехнического устройства высокого уровня сложности, функционирующего в режиме реального времени, требуется решить несколько первостепенных задач, одной из которых является разработка методов и алгоритмов пространственной навигации, локализации и картографирования. Тем не менее, данная задача представляется наиболее трудной для программно-аппаратной реализации.

В данной работе рассматриваются параметры автоматизированной системы управления прототипом мобильной платформы как объект автоматизации. Основной целью работы стали:

- 1) изучение принципов работы алгоритмов программного комплекса пространственной навигации и мониторинга;
- 2) разработка автоматизированной системы регулирования параметров мобильной навигационной платформы, базирующейся на датчиках визуальной информации и датчиков типа лидар;

3) синтез модели мобильного робота на основе алгоритмов SLAM-навигации.

Для реализации цели были поставлены следующие задачи:

- анализ алгоритмов SLAM-навигации;
- освоение навыков работы с основными SLAM алгоритмами программного комплекса пространственной навигации и мониторинга;
- выбор структуры АСУ;
- разработка структурной схемы РУ;
- разработка структурной схемы подсистемы навигации;
- разработка подсистемы телеуправления;
- сравнение работы методов PTAM и ORB-SLAM по качеству визуальной одометрии;
- реализация и сравнение работы алгоритмов SLAM-навигации основанные на использовании лазерных дальномеров;
- интеграция фреймворка Google Cartographer с алгоритмами Obstacle Avoidance;
- методы аппаратной компоновки для мобильных роботов для решения задач SLAM-навигации.

1 Анализ литературных источников

1.1 Алгоритмы VSLAM

Одновременная локализация и картографирование (SLAM) - это метод построения 3D-структуры области окружающего пространства, получаемой при перемещении датчика в окружающей среде. Изначально данная группа методов разрабатывалась для автономного управления роботами в робототехнике [1]. Однако сейчас многие прикладные программы, основанные на SLAM-технологии, такие как онлайн-3D-моделирование на основе компьютерного зрения, визуализация на основе дополненной реальности (AR) и автомобили с беспилотным управлением, получили широкое распространение. В ранних алгоритмах SLAM было интегрировано множество различных типов датчиков, таких как лазерные дальномеры, энкодеры, инерционные датчики, GPS и камеры. В последние годы активно обсуждается SLAM, использующий только камеры, потому что конфигурация датчика проста, но технические трудности реализации выше, чем у других алгоритмов. Поскольку вход такого SLAM представляется только визуальной информацией, данный метод здесь упоминается как визуальный SLAM (VSLAM). Алгоритмы VSLAM широко применяются в области компьютерного зрения, робототехники и AR [6]. В частности, они подходят для оценки положения камеры в системах AR, так как конфигурация систем, например, на планшетах или смартфонах, оборудованных камерой, может быть довольно простой. Одним из важных требований в AR-системах является реакция в реальном времени на плавное и интерактивное объединение реальных и виртуальных объектов. Для достижения ответа на легком портативном устройстве с ограниченным вычислительным ресурсом были предложены различные алгоритмы VSLAM с низким уровнем вычислительных затрат. Применение таких алгоритмов VSLAM не ограничивается системами AR. Например, они также полезны в

беспилотных автономных транспортных средств и в робототехнике [7]. Несмотря на то, что алгоритмы VSLAM были предложены для разных целей в разных исследовательских сообществах, в основном они разделяют общие части основных технических идей, и каждый из них может использоваться для достижения разных целей. Поэтому следующие алгоритмы требуется классифицировать и обобщить в данной исследовательской работе.

В работе рассматриваются алгоритмы VSLAM реального времени, которые заметно развиваются с 2010. В общем, технические трудности VSLAM выше, чем у SLAM на основе других датчиков, потому что камеры могут получать меньше входной визуальной информации из ограниченного поля видимости по сравнению с 360° лазерным измерением, которое обычно используется в робототехнике. Для такой входной информации необходимо последовательно оценивать положение камеры и одновременно восстанавливать трехмерную структуру неизвестной среды. В 2000-х годах, ранняя реализация VSLAM с использованием монокулярной камеры основывалась на отслеживании и отображении особых точек. Это называется «характеристический подход». Чтобы справиться с безмасштабированными средами или средами без особых точек, был предложен VSLAM для локализации и картографирования, работающий непосредственно со всем изображением без обнаружения особых точек. Это называется «прямым подходом». С появлением недорогих RGB-D-датчиков, таких как Microsoft Kinect, были предложены алгоритмы VSLAM с монокулярным изображением и его глубиной. Таким образом, существующие алгоритмы VSLAM, представленные в этой работе, классифицируются в соответствии с характеристическим, прямым и RGB-D подходами.

1.2 Элементы VSLAM

Сначала следует представить базовую структуру, которой следуют большинство алгоритмов VSLAM с конца 2000-х годов.

1.3 Основные модули

Структура в основном состоит из трех модулей:

- 1) инициализация;
- 2) отслеживание;
- 3) картографирование.

Чтобы начать VSLAM, для оценки положения камеры и 3D-реконструкции в неизвестной среде, необходимо задать определенную систему координат. Поэтому при инициализации сначала должна быть определена глобальная система координат, а часть среды будет восстановлена как исходная карта в этой системе координат. После инициализации локализация и картографирование выполняются для непрерывной оценки позиции камеры. При локализации, восстановленная карта отслеживается на изображении для оценки положения камеры относительно карты. Для этого 2D-3D соответствия между изображением и картой сначала получаются из соответствия функций или отслеживания признаков на изображении. Затем позиция камеры вычисляется из соответствий путем решения задачи Perspective-n-Point (PnP) [10, 11]. Следует отметить, что большинство алгоритмов VSLAM предполагают, что параметры встроенной камеры предварительно откалиброваны, таким образом они должны быть известны. Следовательно, позиция камеры обычно эквивалентна внешним параметрам камеры при перемещении и вращении в глобальной системе координат. При картографировании карта расширяется путем вычисления трехмерной структуры среды, когда камера охватывает

новые неизвестные области, где картографирование еще, собственно, не выполнено.

1.4 Дополнительные модули для увеличения стабильности и точности VSLAM

Следующие два дополнительных модуля также включены в алгоритмы VSLAM в соответствии с целями применения:

- 1) релокализация;
- 2) оптимизация глобальной карты.

Релокализация происходит вследствие быстрого, скачкообразного движения камеры либо большого количества помех во входной информации, когда вычисления собственного перемещения перестают выполняться. В этом случае необходимо снова вычислить положение камеры относительно карты. Поэтому этот процесс называется «релокализация». Если в системах VSLAM релокализация не включена, системы перестают работать после того, как процесс отслеживания будет прерван, и такие системы практически бесполезны. Поэтому быстрый и эффективный метод релокации до сих пор обсуждается в литературе. Обратите внимание, что это также называют проблемой «похищенных роботов» в робототехнике.

Карта обычно включает в себя ошибку суммарной оценки в соответствии с расстоянием перемещения камеры. Чтобы подавить ошибку, обычно выполняется глобальная оптимизация. В этом процессе карта уточняется с учетом последовательности всей информации о карте. Когда карта перестраивается, так что начальная область снова захватывается после некоторого движения камеры, можно вычислить дополнительные данные, которые и представляют собой накопленную ошибку от начала до текущего момента времени. Затем ограничение цикла из дополнительных данных

используется как ограничение для подавления ошибки в глобальной оптимизации.

Замыкание петли - это метод получения дополнительных данных. При замыкании петли сначала выполняется поиск уже замкнутой петли путем сопоставления текущего изображения с ранее полученными изображениями. Если петля обнаружена, это означает, что камера захватывает один из ранее наблюдаемых кадров. В этом случае, во время движения камеры, может быть оценена накопленная ошибка. Обратите внимание, что процедура обнаружения замкнутых петель может быть выполнена с использованием тех же методов, что и релокализация. В основном, релокализация выполняется для восстановления положения камеры, а обнаружение петли выполняется для получения геометрической последовательности карты.

Оптимизация графа положения широко использовалась для подавления накопленной ошибки путем оптимизации позиции камеры [12, 13]. В этом методе связь между позициями камеры представлена в виде графика, а последовательный граф построен для подавления ошибки в оптимизации. *Bundle Adjustment* (BA) также используется для минимизации ошибки повторения изображения на карте, оптимизируя положение карты и камеры [14]. В больших средах эта процедура оптимизации используется для минимизации ошибок оценки. В небольших средах BA может выполняться без замыкания петли, поскольку накопленная ошибка мала.

1.5 Связанные технологии

VSLAM, визуальная одометрия и интерактивная структура от движения предназначены для оценки движения камеры и трехмерной структуры в неизвестной среде. В этом разделе рассматривается взаимосвязь между ними.

1.5.1 Визуальная одометрия

Одометрия - это оценка последовательных изменений положения во времени с использованием одометрических датчиков, таких как колесный энкодер, для получения относительного перемещения датчика визуальной информации. Одометрия с камеры, называемая визуальной одометрией (VO), также является одной из активных областей исследований. С технической точки зрения, VSLAM и VO - очень важные методы, потому что в основе обоих методов лежит оценивание положения датчика. Согласно исследованиям в робототехнике [18, 19], связь между VSLAM и VO может быть представлена следующим образом.

$$\text{VSLAM} = \text{VO} + \text{оптимизация глобальной карты}$$

Основное различие между этими двумя методами - глобальная оптимизация карты в картографировании. В VO геометрическая последовательность карты рассматривается только в небольшой части карты или вычисляется только относительное движение камеры без картографирования. С другой стороны, в VSLAM обычно рассматривается глобальная геометрическая последовательность карты. Поэтому для построения геометрически последовательной карты, глобальная оптимизация выполняется в последних алгоритмах VSLAM.

1.5.2 Структура от движения

Структура из движения (SfM) - это метод оценки движения камеры и трехмерной структуры среды периодическим способом [24]. На сегодняшний день был предложен метод SfM, который работает в режиме онлайн. Авторы назвали его SfM реального времени. С технической точки зрения нет окончательной разницы между VSLAM и SfM реального времени. Это может

быть поэтому, что слово «SfM реального времени» не встречается в последних статьях.

Таким образом, VSLAM, VO и SfM в режиме реального времени имеют множество общих компонентов. Поэтому, для структурного обобщения, данные технологии не различаются в данной работе.

2 Анализ алгоритмов SLAM-навигации

2.1 Функциональные методы

В литературе существуют два типа характеристических методов: основанные на фильтрах и методы основанные на ВА. В этом разделе сравниваются оба метода. Несмотря на то, что некоторые из методов были предложены до 2010 года, их можно рассматривать как фундаментальные структуры для других методов.

2.1.1 MonoSLAM

Первый монокулярный vSLAM был разработан в 2003 году Дэвисоном и др. [26, 27]. Они назвали его MonoSLAM. MonoSLAM рассматривается как типичный метод в алгоритмах vSLAM на основе фильтра. В MonoSLAM движения камеры и трехмерной структуры неизвестной среды оценивается одновременно с использованием расширенного фильтра Калмана (EKF). Шесть степеней свободы камеры (DOF) и трехмерные положения особых точек представлены в виде вектора состояния в EKF. Равномерное движение принимается в модели прогнозирования, а результат отслеживания точки объекта используется в качестве наблюдения. В зависимости от движения камеры к вектору состояния добавляются новые особые точки. Обратите внимание, что начальная карта создается путем наблюдения за известным объектом, где определена глобальная система координат. Таким образом, MonoSLAM состоит из следующих компонентов:

- 1) инициализация карты выполняется с использованием известного объекта;
- 2) движение камеры и трехмерные позиции точек объекта оцениваются с использованием EKF.

Проблема этого метода - это вычислительная стоимость, которая увеличивается пропорционально размеру среды. В больших средах размер вектора состояния становится большим, поскольку количество особых точек велико. В этом случае трудно производить вычисления в реальном времени.

2.1.2 PTAM

Чтобы решить проблему вычислительной стоимости в MonoSLAM, PTAM [15] разделил отслеживание и картографирование на разные потоки CPU. Эти два потока выполняются параллельно, так что вычислительная стоимость картографирования не влияет на отслеживание. В результате в картографировании может использоваться *BA*, требующая вычислительных затрат при оптимизации. Это означает, что отслеживание оценивает движение камеры в реальном времени, а картографирование оценивает точные трехмерные позиции точек с вычислительной стоимостью. PTAM - это первый метод, который включает *BA* в алгоритмы vSLAM реального времени. После публикации PTAM большинство алгоритмов vSLAM используют этот тип многопоточных подходов.

В PTAM исходная карта восстанавливается с использованием пятиточечного алгоритма [28]. При отслеживании отображаемые точки проецируются на изображение, для создания 2D-3D соответствия, используя сопоставление текстур. Из соответствий могут быть вычислены положения камеры. При сопоставлении трехмерные позиции новых точек функции вычисляются с использованием триангуляции в определенных кадрах, называемых ключевыми кадрами. Одним из значительных вкладов PTAM является внедрение картографирования на основе ключевого кадра в VSLAM. Входной кадр выбирается как ключевой кадр, при измерении большого несоответствия между входным кадром и одним из ключевых кадров. Для точной триангуляции требуется большая диспропорция. В

отличие от MonoSLAM, 3D-позиции особых точек оптимизируются с использованием *Lobal BA* с некоторыми ключевыми кадрами, и глобальным *BA* со всеми ключевыми кадрами на карте. Кроме того, в процессе отслеживания более новое видение PTAM использует алгоритм релокализации [29]. Он использует рандомизированный древовидный классификатор свойств для поиска ближайшего ключевого кадра входного изображения. Таким образом, PTAM состоит из следующих четырех компонентов:

- 1) инициализация карты выполняется с помощью пятиточечного алгоритма;
- 2) положения камеры оцениваются по совпадающим особым точкам между точками карты и входным изображением;
- 3) трехмерные позиции точек объекта оцениваются путем триангуляции, а оцененные 3D-позиции оптимизируются *BA*;
- 4) процесс отслеживания восстанавливается рандомизированным древовидным поиском.

По сравнению с MonoSLAM, в PTAM, система может обрабатывать тысячи особых точек, разделяя отслеживание и картографирование на разные потоки в процессоре.

Было предложено множество расширенных алгоритмов PTAM. Команда разработчиков Castle разработала многоплановую версию PTAM [30]. Klein разработала версию PTAM для мобильного телефона [31]. Для запуска PTAM на мобильных телефонах, было уменьшено разрешение входного изображения, количество точек карты и ключевых кадров. Кроме того, они учитывают искажение поворотного затвора в *BA* для получения точного результата оценки, потому что поворотный затвор, из-за его дешевой стоимости, обычно устанавливается на большинстве мобильных камер. Поскольку PTAM может восстанавливать только разреженную трехмерную

структуру среды, третий поток можно использовать для восстановления плотной трехмерной структуры среды [32, 33].

2.1.3 Сравнение MonoSLAM и PTAM

Различие между картографированием на основе EKF в MonoSLAM и картографированием на основе ВА с ключевыми кадрами в PTAM обсуждалось в работе [34]. Согласно источнику, для повышения точности vSLAM важно увеличить количество точек на карте. С этой точки зрения подход на основе ВА лучше, чем подход на основе EKF, поскольку он позволяет обрабатывать большое количество точек.

2.1.4 Методы оптимизации глобальной карты

Геометрическая последовательность всей карты поддерживается с использованием ВА для ключевых кадров, как описано выше. Однако, в общем, ВА подвержен проблеме локального минимума из-за большого количества параметров, включая положения камеры ключевых кадров и точки на карте. Оптимизация графа положения - это решение, позволяющее избежать этой проблемы при замыкании петли, как описано в разделе 2. При замыкании петли сначала оптимизируются положения камеры, используя ограничение цикла. После оптимизации положений камеры выполняется ВА для оптимизации как трехмерных позиций точек объекта, так и положения камеры. Для замыкания петли используется подход, основанный на визуальной информации [35]. Они использовали метод поиска изображений на основе «bag-of-words», чтобы обнаружить один из ключевых кадров, изображение которого аналогично текущему кадру [36].

В системе VSLAM [35] в качестве визуального датчика выбирается стереокамера. В этом случае шкала системы координат является

фиксированной и известной. Однако в монокулярных случаях VSLAM существует неопределенность масштаба, и масштаб может изменяться во время движения камеры, если глобальная *BA* не выполняется. В этом случае возникает проблема смещения масштаба, и масштаб системы координат в каждом кадре может не совпадать. Чтобы исправить смещение масштаба, положение камеры следует оптимизировать с 7 степенями свободы. Разработчики из *Strasdat Group* [37] предложили метод оптимизации позиций камеры с 7 степенями свободы на основе преобразования подобия.

В качестве расширения PTAM, ORB-SLAM включает в себя *BA*, визуальное обнаружение замкнутых петель и оптимизацию графа положения с 7 степенями свободы. Таким образом, ORB-SLAM является самой полной функциональной монокулярной системой VSLAM. ORB-SLAM распространяется на стерео VSLAM и RGB-D VSLAM [39].

2.2 Прямые методы

В отличие от функциональных методов в предыдущем разделе, прямые методы напрямую используют входное изображение без какой-либо абстракции с использованием рукописных функциональных детекторов и дескрипторов. В общем, фотометрическая последовательность используется как погрешность измерения в прямых методах, тогда как геометрическая последовательность, такая как положение особых точек на изображении, используется в характеристических методах. В этом разделе мы представим некоторые ведущие прямые методы.

2.2.1 DTAM

Newcombe предложил полностью прямой метод [43], называемый DTAM. В DTAM отслеживание выполняется путем сравнения входного

изображения с синтетическими изображениями кадров, сгенерированными из восстановленной карты. Это просто эквивалентно регистрации между изображением и 3D-моделью карты и эффективно реализуется на графическом процессоре в DTAM. Картографирование выполняется с использованием многоосевого стерео [44], а затем карта оптимизируется с учетом непрерывности пространства [45], таким образом, чтобы можно было вычислить трехмерные координаты всех пикселей. Первоначальная карта глубины создается с использованием стереометрического измерения, такого как PTAM. Таким образом, DTAM состоит из следующих трех компонентов:

- 1) инициализация карты производится стереометрическим измерением;
- 2) движение камеры оценивается с помощью синтетической генерации кадров с восстановленной карты;
- 3) информация о глубине оценивается для каждого пикселя с использованием многоосевого стерео, а затем оптимизируется с учетом непрерывности пространства.

Алгоритм DTAM оптимизирован для обеспечения обработки в реальном времени на мобильных телефонах [46]. В принципе, эти методы [43, 46, 47] предназначены для быстрого и онлайн-3D-моделирования.

Следует отметить, что группа программистов *Stühmer* ранее предложили вариационный подход для оценки информации о глубине для каждого пикселя [47]. Они используют аналогичную весовую функцию для отображения как и в DTAM. Однако, в этом методе PTAM [15] использовался для отслеживания.

2.2.2 LSD-SLAM

LSD-SLAM является другим ведущим методом среди прямых методов. Основная идея LSD-SLAM вытекает из идеи полуплотной VO [20]. В этом методе восстанавливаемые объекты ограничены областями с градиентом

яркости по сравнению с DTAM, который восстанавливает полные области. Это означает, что он игнорирует области без текстур, поскольку трудно оценить точную информацию о глубине по изображению. При сопоставлении, начальные значения глубины для каждого пикселя сначала устанавливаются как случайные значения, а затем эти значения оптимизируются на основе фотометрической последовательности. Поскольку этот метод не учитывает геометрическую последовательность всей карты, этот метод называют визуальной одометрией.

В 2014 году полуплотная VO была распространена в LSD-SLAM [21]. В LSD-SLAM обнаружение замыкания петли и оптимизаций графа положения с 7 степенями свободы, как описано в предыдущих разделах, прибавляется к алгоритму полуплотной визуальной одометрии [20]. Таким образом, LSD-SLAM состоит из следующих четырех компонентов:

- 1) случайные значения устанавливаются как начальные значения глубины для каждого пикселя;
- 2) движение камеры оценивается с помощью синтетической генерации изображений с восстановленной карты;
- 3) реконструированные области ограничены областями градиента высокой яркости;
- 4) для создания геометрически последовательной карты используется оптимизация геометрии 7 DOF.

В принципе, эти полуплотные подходы могут обеспечить обработку в реальном времени на процессоре. Кроме того, они оптимизировали алгоритм LSD-SLAM для мобильных телефонов, учитывая их архитектуру ЦП. В работе [48] они также оценивали точность алгоритма LSD-SLAM для входных изображений с низким разрешением. LSD-SLAM распространяется на стереокамеры и однонаправленные камеры [49, 50].

2.2.3 SVO и DSO

Команда разработчиков *Forster Group* предложили полупрямую VO (SVO) [51]. Хотя отслеживание выполняется с помощью сопоставления точек, сопоставление выполняется прямым методом. В функциональных методах для определения соответствий используются дескрипторы функций и трекер Lucas-Kanade [52]. В отличие от функциональных методов, движение камеры оценивается путем минимизации фотометрических ошибок, связанных с объектами. Этот метод можно рассматривать как редкую версию DTAM и LSD-SLAM.

Недавно разработчики из *Engel Group* предложили прямую разреженную одометрию (DSO) [53]. В отличие от SVO, DSO является полностью прямым методом. Чтобы подавить накапливаемую ошибку, DSO максимально разделяет факторы ошибок с геометрической и фотометрической точек зрения. В DSO входное изображение делится на несколько блоков, а затем в качестве кандидатов на реконструкцию выбираются точки высокой яркости. Используя эту стратегию, точки распространяются по всему изображению. Кроме того, для достижения высокоточной оценки DSO использует как геометрические, так и фотометрические результаты калибровки камеры. Следует отметить, что DSO рассматривает только локальную геометрическую последовательность. Поэтому DSO классифицируется как VO, а не vSLAM.

2.3 RGB-D vSLAM

В последнее время RGB-D камеры, использующие структурированный свет, такие как Microsoft Kinect, становятся дешевыми и маленькими. Поскольку такие камеры дают трехмерную информацию в режиме реального времени, эти камеры также используются в алгоритмах vSLAM.

2.3.1. Разница с монокулярным vSLAM

Используя камеры RGB-D, можно получить напрямую трехмерную структуру среды с ее текстурной информацией. Кроме того, в отличие от монокулярных алгоритмов VSLAM, масштаб системы координат известен, поскольку трехмерную структуру можно получить в метрическом пространстве.

Базовая структура VSLAM на основе глубины (D) заключается в следующем. Для оценки движения камеры широко использовался итеративный алгоритм ближайших точек (ICP). Затем трехмерная структура среды восстанавливается путем объединения нескольких карт глубины. Чтобы включить RGB в VSLAM на основе глубины, было предложено много подходов.

Следует отметить, что большинство пользовательских камер для измерения глубины разработаны для использования в помещениях. Они проецируют ИК-шаблоны на окружающую среду для измерения глубины. Трудно обнаружить испускаемые ИК-изображения вне помещений. Кроме того, существует ограничение в диапазоне измерения глубины, поскольку RGB-D датчики могут охватывать окружающую среду в диапазоне от 1 до 4 м.

2.3.2 KinectFusion

Разработчики из *Newcombe Group* предложили KinectFusion в 2011 году. В KinectFusion используется пространство вокселей для представления трехмерной структуры среды. Трехмерная структура среды реконструируется путем объединения полученных карт глубины в воксельном пространстве, а движение камеры оценивается по алгоритму ICP с использованием уже оцененной трехмерной структуры и входной карты глубины, которая

является VSLAM основанным на использовании данных глубины. KinectFusion реализован на GPU для достижения обработки в реальном времени.

Разработчики из *Kahler Group* реализовали обработку KinectFusion в реальном времени на мобильных устройствах. Чтобы уменьшить вычислительную стоимость, в процессе картографирования они используют хеширование воксельного блока. RGB-D vSLAM обрабатывает огромный объем данных, что тормозит скорость работы алгоритма. Уменьшение количества данных происходит путем объединения копланарных точек.

2.3.3 SLAM ++

Команда *Salas-Moreno* предложил алгоритм VSLAM RGB-D на уровне объектов. В этом методе в базу данных регистрируются несколько 3D-объектов, и эти объекты распознаются в режиме онлайн. Оценочная карта уточняется, путём распознавания 3D-объектов, где 3D-точки заменяются 3D-объектами, чтобы уменьшить объем данных.

В качестве аналогичного алгоритма команда разработчиков *Tateno* предложила для RGB-D SLAM метод сегментации в реальном времени. Сегментированные объекты помечаются, а затем данные объекты могут использоваться в качестве распознаваемых целей.

2.3.4 Методы RGB-DVO и глобальной оптимизации карт

Для отслеживания, RGB изображения также используются в алгоритмах RGB-D vSLAM. Относительное движение камеры оценивается путем отслеживания особых точек между последовательными кадрами. Матрица перехода затем оценивается с использованием отслеживаемых особых точек, и эта матрица перехода уточняется с помощью алгоритма ICP

с использованием карт глубины. С другой стороны, были предложены методы отслеживания движения камеры на основе фотометрической последовательности [22, 23, 64]. Это фотометрическое отслеживание движения камеры, также используется для создания плотных методов vSLAM на основе монокулярных камер [20, 21, 43].

Для получения геометрически последовательной карты в алгоритмах RGB-D VSLAM используются оптимизация графа положения и оптимизация графа деформации. В *Kerl Group* использовали оптимизацию графа положения для уменьшения накапливаемой ошибки [23]. Эта оптимизация графа положения почти такая же, как замыкание петли в монокулярных алгоритмах VSLAM. Программисты в *Whelan Group* использовали оптимизацию графа положения для оптимизации движения камеры и оптимизации графика деформации для уточнения карты, соответственно. В отличие от других работ [23], расчетная карта также уточняется. Оптимизация графа деформации часто используется для определенных кадров, а движение камеры оценивается путем сопоставления изображения RGB-D с восстановленной моделью. Они показали геометрически последовательную модель, которая может быть получена с использованием оптимизации графа деформации как можно чаще.

Обратите внимание, что API-интерфейсы RGB-D SLAM предоставляются в таких пользовательских устройствах, как *Google Tango2* и *Structure Sensor3*. В частности, *Google Tango* обеспечивает стабильный результат оценки путем объединения всей информации с датчика.

2.4 Сравнение и анализ

Как указано выше, структура алгоритмов VSLAM состоит из пяти модулей: инициализация, локализация, картографирование, релокализация и глобальная оптимизация карт. Поскольку каждый алгоритм vSLAM

использует разные методологии для каждого модуля, особенности алгоритма vSLAM сильно зависят от использования данных методологий. Поэтому важно понимать каждый модуль алгоритма vSLAM, чтобы знать его производительность, преимущества и ограничения.

Следует отметить, что отслеживание и картографирование (**TAM**) используется вместо использования локализации и картографирования. **TAM** впервые использовался в *Parallel Tracking and Mapping* (PTAM) [15], поскольку локализация и картографирование не выполняются одновременно в традиционном способе. Отслеживание выполняется в каждом кадре с одним потоком, тогда как картографирование выполняется в определенный момент времени с другим потоком. После того, как был предложен PTAM, большинство алгоритмов VSLAM наследуют структуру *TAM*. Поэтому в этой работе рассматривается *TAM*.

На рисунке 2.1 приведено краткое описание функциональных методов. MonoSLAM был разработан в 2003 году [26]. Как отслеживание, так и картографирование реализуется последовательно и одновременно с использованием EKF. PTAM был разработан в 2007 году [15]. Они предложили разделить отслеживание и картографирование на разные потоки на процессоре. Этот многопоточный подход позволяет обрабатывать тысячи точек на карте. В большой окружающей среде трудно достичь оптимального результата при отображении глобальной карты и положения камеры из-за проблемы локального минимума в *BA*. Чтобы избежать этой проблемы, перед *BA* можно использовать обнаружение замкнутой петли и оптимизацию графа положения. ORB-SLAM [38] включает в себя многопоточное отслеживание, сопоставление и обнаружение замкнутой петли, а карта оптимизирована с использованием графа положения и *BA*, и это можно рассматривать как пакет моноблочных VSLAM «все-в-одном». Поскольку ORB-SLAM является проектом с открытым исходным кодом, мы можем легко использовать всю систему VSLAM в нашей локальной среде.

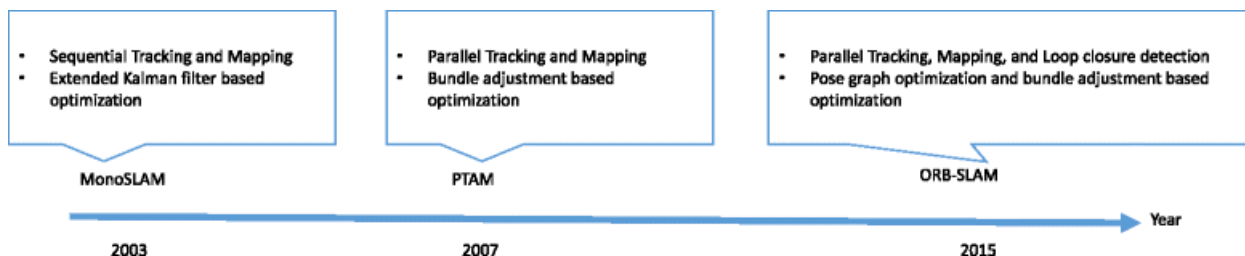


Рисунок 2.1 - Краткое описание функциональных методов

Функциональные алгоритмы VSLAM, основанные на точках, обычно используют рукописные детекторы и дескрипторы функций и могут обеспечить стабильные результаты оценки в хорошо текстурированных средах. Тем не менее, трудно обрабатывать изогнутые края и другие сложные формы, используя такие функции. В некоторых особых случаях, таких как плохо текстурированная среда, линейные функции использовались в качестве функций изображения [40, 41]. Кроме того, функциональные точки и кромки объединяются для обеспечения надежной оценки по отношению к размытому входному изображению [42].

Прямые методы можно классифицировать по плотности карт. Плотные методы [43, 47] создают плотную карту, рассчитанную таким образом, что значения глубины оцениваются для каждого пикселя в каждом ключевом кадре. Эти методы могут быть полезны для 3D-моделирования в реальном времени с использованием графического процессора.

В отличие от плотных методов, полуплотные [21] и редкие методы фокусируются на приложениях, основанных на отслеживании положения датчиков. Эти методы могут выполняться в режиме реального времени на центральных процессорах.

В ходе исследования были представлены последние алгоритмы VSLAM в ключевой период с 2010 по 2018 год. В основном, все алгоритмы VSLAM состоят из инициализации, оценки движения камеры, оценки 3D-структуры, глобальной оптимизации и релокализации. В последнее время «прямые методы» являются активным полем исследований в монокулярном VSLAM. RGB-D VSLAM был разработан также в последние годы, так как

многие пользовательский RGB-D камеры сейчас можно приобрести по низкой цене. Несмотря на то, что алгоритмы vSLAM разрабатывались с 2003 года, vSLAM по-прежнему является активным полем исследований.

В таблице 2.1 приведено краткое сравнение представленных методов. Каждый алгоритм имеет разные характеристики. Необходимо выбирать соответствующий алгоритм, рассматривая цель его использования.

Таблица 2.1 - Сравнение представленных алгоритмов

Метод	Тип метода	Плотность карты	Глобальная оптимизация	Замыкание петель
Моно-SLAM	Функциональный	Разреженная	Нет	Нет
PTAM	Функциональный	Разреженная	Есть	Нет
ORB-SLAM	Функциональный	Разреженная	Есть	Есть
DTAM	Прямой	Плотная	Нет	Нет
LSD-SLAM	Прямой	Полу-плотная	Есть	Есть
SVO	Полу-прямой	Разреженная	Нет	Нет
DSO	Прямой	Разреженная	Нет	Нет
KinectFusion	RGB-D	Плотная	Нет	Нет
Dense visual SLAM	RGB-D	Плотная	Есть	Есть
ElasticFusion	RGB-D	Плотная	Есть	Есть
SLAM++	RGB-D	Плотная	Есть	Есть

В этой работе были рассмотрены последние алгоритмы vSLAM, основанные только на использовании камеры. С другой стороны, алгоритмы SLAM, которые используют визуальные и инерциальные данные, называются визуально-инерциальными SLAM. Объединяя визуальные и инерциальные данные, мы можем получить более стабильные результаты оценки. Кроме того, возможно использование детальной информации о датчике для решения оценки масштаба и компенсации искажения затвора. В настоящее время практически все смартфоны и планшетные устройства

имеют камеры, GPS, гироскоп и акселерометр. Таким образом, считается, что сопряжение датчиков - одно из направлений для реализации надежных и практичных систем VSLAM.

3 Современные подходы к решению задачи SLAM для лазерных дальномеров. LOAM

Картографирование при помощи лидаров является распространённым явлением, поскольку лидары могут обеспечить высокую частоту измерения дальности, где погрешности относительно постоянны независимо от измеренных расстояний. В случае, когда единственным движением лидара является вращение лазерного луча, регистрация облака точек проста.

Однако, если сам лидар движется, как во многих областях применения, точное картографирование требует измерения положения лидара во время непрерывного определения дальности. Одним из распространенных способов решения этой проблемы является использование независимой оценки положения (например, с помощью GPS / INS) для регистрации лазерных точек в фиксированной системе координат. Другой набор методов использует измерения одометрии, или системы визуальной одометрии для регистрации лазерных точек.

Поскольку одометрия объединяет небольшие инкрементные движения во времени, она связана с дрейфом, и большое внимание уделяется уменьшению дрейфа (например, с помощью замыкания петли). Здесь рассмотрим случай создания карт с одометрией с низким смещением с использованием двухосевого лидара, движущегося в 6-DOF. Основным преимуществом использования лидара является его нечувствительность к окружающему освещению и оптической текстуре в сцене.

Последние разработки лидаров позволили уменьшить их размеры и вес. Лидары могут использоваться человеком, который исследует окружающую среду, или же быть прикреплен к транспортному средству. Поскольку LOAM метод предназначен для решения проблем, связанных с минимизацией дрейфа при оценке одометрии, в настоящее время он не включает “замыкание петель”.

Этот метод учитывает сложность как с низким дрейфом, так и с низкой вычислительной сложностью без необходимости измерения с высокой точностью или инерциальных измерений. Ключевой идеей при получении этого уровня производительности является разделение типично сложной проблемы одновременной локализации и отображения (SLAM) [8], которая стремится оптимизировать большое количество переменных, одновременно, двумя алгоритмами. Один алгоритм выполняет одометрию на высокой частоте, но с низкой точностью, чтобы оценить скорость лидара.

Другой алгоритм работает с частотой на порядок ниже для точного сопоставления и регистрации облака точек. Несмотря на то, что в этом нет необходимости, если имеется IMU, может быть предоставлено предварительное движение, чтобы помочь учесть высокочастотное движение. В частности, оба алгоритма выделяют характерные точки, расположенные на острых кромках и плоских поверхностях, и сопоставляют характерные точки с сегментами линий кромок и плоскими участками поверхности, соответственно.

В алгоритме одометрии соответствия характерных точек находятся благодаря быстрому вычислению. В алгоритме картографирования соответствия определяются путем изучения геометрического распределения локальных точечных кластеров через соответствующие собственные значения и собственные векторы.

Разбивая исходную проблему, вначале решается более простая задача - оценка движения в режиме реального времени. После чего отображение выполняется как пакетная оптимизация (аналогично методам итеративной ближайшей точки (ICP) для получения высокоточных оценок движения и карт. Структура параллельного алгоритма обеспечивает возможность решения проблемы в режиме реального времени. Кроме того, поскольку оценка движения проводится на более высокой частоте, для сопоставления дается достаточно времени в целях обеспечения точности. При работе на

более низкой частоте алгоритм картографирования может включать в себя большое количество особых точек и использовать достаточно много итераций для сходимости.

3.1 Базовые отличия современного метода LOAM

Когда скорость сканирования лидара высока по сравнению с его внешним движением, искажением движения внутри сканирования часто можно пренебречь. В этом случае стандартные методы ICP могут использоваться для согласования лазерных сканов между различными сканированиями.

Кроме того, предлагается двухэтапный метод удаления искажения: за этапом оценки скорости на основе ICP следует этап компенсации искажения с использованием вычисленной скорости. Подобный метод также используется для компенсации искажений, вносимых одноосным трехмерным лидаром. Однако, если движение сканирования относительно медленное, искажение движения может быть серьезным. Это особенно актуально, когда используется двухосевой лидар, поскольку одна ось обычно намного медленнее другой. Часто для измерения скорости используются другие датчики, с помощью которых можно устранить искажение. Например, лидарное облако может быть зарегистрировано путем оценки состояния по визуальной одометрии, интегрированной с IMU. Когда одновременно доступно несколько датчиков, таких как GPS / INS и колесные датчики, проблема обычно решается с помощью расширенного фильтра Калмана или фильтра частиц. Эти методы могут создавать карты в режиме реального времени, чтобы помочь планировщику пути и предотвращению столкновений при навигации робота.

Если 2-осевой лидар используется без помощи других датчиков, оценка движения и коррекция искажений становятся одной из проблем. Метод,

используемый командой разработчиков Barfoot состоит в том, чтобы создавать визуальные изображения из результатов лазерной интенсивности, и сопоставлять визуально различимые особенности между изображениями, чтобы восстановить движение наземного транспортного средства. LOAM метод использует модель линейного движения, но с различными типами функции. LOAM метод извлекает и сопоставляет геометрические элементы в декартовом пространстве и имеет более низкие требования к плотности облаков. Подход, наиболее близкий к LOAM - это подход Боссе и Зилота. Они используют 2-осевой лидар для получения облака точек, которое регистрируется путем сопоставления геометрических структур локальных кластеров точек. Кроме того, они используют несколько 2-осных лидаров, чтобы нанести на карту подземный рудник. Этот метод включает в себя IMU и использует замыкание цикла для создания больших карт. Предложенный теми же авторами, Zebedee представляет собой картографическое устройство, состоящее из двумерного лидара и IMU, прикрепленных к ручному стержню через пружин. Траектория восстанавливается методом пакетной оптимизации, который обрабатывает сегментированные наборы данных с граничными условиями, добавленными между сегментами. В этом методе измерения IMU используются для регистрации лазерных точек, а оптимизация используется для исправления смещений IMU. По сути, методы Боссе и Зилота требуют последующей пакетной обработки для разработки точных карт и, следовательно, не подходят для приложений, где карты необходимы в режиме реального времени. Для сравнения, предлагаемый метод в режиме реального времени создает карты, которые качественно похожи на карты Боссе и Злот. Различие состоит в том, что LOAM метод может предоставить оценки передвижения для управления автономным транспортным средством. Кроме того, метод использует преимущества схемы сканирования лидара и распределения облаков точек. Сопоставление

признаков осуществляется с обеспечением скорости и точности вычислений в алгоритмах одометрии и картографирования соответственно.

4 Методы и проблемы локализации в задачах синтеза мобильных роботов

Процесс определения своего положения в среде называется локализацией. На текущий момент существует два наиболее простых метода вычисления собственного местоположения:

1) одометрия: этот метод основан на простом принципе: если вы знаете начальную позицию, траекторный вектор робототехнического устройства и вектор его скорости, то на основании этой информации можно вычислить текущее местоположение робота. Простыми словами: если вы знаете начальную позицию робота и помните каждое движение, которое он совершил с тех пор, вы можете легко рассчитать его текущую позицию;

2) зондирование: эта техника использует данные из различных сенсорных систем. Данные датчика сравниваются с описанием состояния текущей среды. Этот процесс сравнения позволяет нам определить положение относительно описания хранимой среды.

Оба эти метода имеют свои сильные и слабые стороны, но в обоих подходах есть одна проблема: неопределенность. Мобильные роботы содержат различные датчики и исполнительные механизмы для определения и управления окружающей средой и изменения положения робота. Поскольку практически невозможно создать датчики и исполнительные механизмы, которые не содержат ошибок измерения, то таким образом вычисления перемещения всегда будут содержать в себе ложные данные. Например, если вы заставить робота проехать 1 метр по прямой и впоследствии измерить пройденное расстояние, реальное пройденное расстояние обычно будет отличаться от указанного. Если бы ошибки были одинаковыми каждый раз, было бы легко принять это во внимание. Но так как погрешность между измерениями отличается, никогда нельзя определить, сколько ошибок присутствует в полученных результатах. То же

самое относится и к измерениям датчиков, например, к измерениям расстояния от сонара или лазерных сканеров [54].

При рассмотрении различных типов датчиков, лазерные дальномеры (Lidar), монокулярное зрение и сети Wi-Fi являются популярными методами обнаружения для задач локализации внутри помещений. В последнее время, с развитием компьютерного зрения и обработки изображений, многие разработчики начали использовать в задачах локализации SLAM на основе датчиков визуальной информации. При условии, что захваченные изображения сопоставляются в достаточной степени, объекты могут быть извлечены с использованием масштабно-инвариантного преобразования объектов (SIFT) или метода «Ускоренных функции (SURF)». Другие методы локализации учитывают амплитуду принимаемого сигнала от сетей Wi-Fi. Эти стратегии локализации зависят от предварительно установленных беспроводных аппаратных устройств на сайте и, следовательно, могут быть неприменимы в средах, в которых Wi-Fi запрещен.

Для простейшего модуля локализации можно использовать набор встроенных ультразвуковых датчиков, которые обеспечивают измерения окружающей среды. Однако позиция робота может быть легко потеряна в сложных условиях без помощи информации о кинематике робота. Для обеспечения надежного распознавания изображений, навигации и построения карт был рассмотрен датчик глубины Kinect. Kinect основан на принципе измерения времени дрейфа ИК излучения и может быть использован вне помещений. Поскольку датчики с матрицей глубины способны предоставлять высокоплотные трехмерные данные, вычислительные усилия в реальном времени относительно выше. Кроме того, для хорошо структурированной среды такая аппаратная конфигурация не требуется для позиционирования робота на плоскости.

На текущий момент все большее и большее внимание привлекают датчики типа Lidar. Lidar обладает высокой частотой дискретизации,

высоким угловым разрешением, хорошим качеством измерения дальности до объектов и высокой устойчивостью к изменчивости окружающей среды. В этой работе для локализации используется датчик Lidar Velodyne VLP16.

Анализируя положение элементов в каждом кадре при каждом движении транспортного средства, можно определить расстояние и направление движения транспортного средства. Для разных данных сканирования используется алгоритм итеративной замкнутой точки (ICP), чтобы найти наиболее подходящие условия сопоставления положения робота, включая вращение и перемещение. Однако ICP не всегда может привести к хорошему сопоставлению с точками скана, если проблема регистрации облака точек не решена должным образом. Другими словами, чем лучше регистрация облака точек, тем лучше оценка положения робота. Другой проблемой при применении ICP является эффективность вычислений. Поскольку алгоритм ICP рассматривает правило ближайшей точки для установления соответствия между точками в текущем сканировании и эталонным сканом, сложность поиска соответствия может значительно возрасти, когда сканирование или эталонный скан содержат большие объемы данных.

Были рассмотрены и решены некоторые из проблем, возникающих при применении ICP, в том числе сопоставление ложных точек при больших исходных ошибках, медленная скорость сходимости. Для положения робота, подвергаемого большому начальному угловому смещению, рекомендуется использовать метод итеративного двойного соответствия (IDC). Тем не менее, он требует более высоких вычислений из-за процесса двойного сопоставления сканов. ICP на основе метрик (MbICP) учитывает геометрическое расстояние, которое возникает при одновременно перемещении и вращении. Соответствие между сканами устанавливается с помощью этой меры, и минимизация ошибки осуществляется с точки зрения этого расстояния. MbICP показывает превосходную надежность в случае

существующего большого углового смещения. Среди различных стратегий согласования плоского сканирования можно выделить преобразование с нормальным распределением (NDT) и двухточечный ICP (PLICP) демонстрируют приемлемые характеристики точности сопоставления сканов. NDT преобразует сканы в пространство сетки и стремится найти лучшее соответствие, максимизируя нормальное распределение в каждой ячейке. NDT не требует двухточечной регистрации, поэтому он повышает скорость сопоставления. Однако выбор размера сетки преобладает над оценкой стабильности. PLICP учитывает информацию от геометрической поверхности окружающей среды и пытается минимизировать расстояние, проецируемое на вектор нормали поверхности. Кроме того, решение с учётом замкнутых контуров дается так, что скорость сходимости может быть значительно улучшена [54].

4.1 Метод функционального извлечения особых признаков

Для извлечения функциональных особенностей окружающего пространства робот перемещается в заданной области пространства, при активном алгоритме локализации. Кроме того, зондирование и оценка положения в режиме реального времени является ключом для более эффективной реализации функции поиска особых признаков. Функция извлечения играет важную роль в сокращении объема данных облака точек для ускорения вычислений.

4.1.1 Основная концепция извлечения признаков

Из-за высокого разрешения сканирования Lidar, построение KD-Tree является очень трудоемким процессом, если все точки данных будут введены

в алгоритм ICP. Поэтому извлечение информативных характерных точек для представления окружающей среды является одной из важных задач.

В этом исследовании предлагается процедура сопоставления сканирования признаков. Во-первых, все точки сканирования разделяются на множество кластеров с помощью адаптивного детектора точек останова (ABD). Основная идея ABD состоит в том, чтобы находить особые точки в каждом сканировании, где направление сканирования Lidar выполняется против часовой стрелки и непрерывно. Поэтому, обнаруживая особые точки скана, алгоритм может определить, существует ли разрыв между двумя последовательными точками сканирования. Однако пороговое значение для определения точки останова должно быть адаптивным по отношению к расстоянию сканирования, которое представлено в следующем подразделе.

4.1.2 Новый метод поиска кластеров

Учтите, что для каждого сканирования существует n лучей. Радиус для каждого луча обозначается как r_i , где $i = 1, \dots, n$. Для повышения производительности кластеризации разработан простой и понятный алгоритм адаптивной радиус-кластеризации (ARC):

$$S = R \times \Delta\theta = R \times (\alpha \times 2\pi/360), \quad R = \min(r_i, r_{i-1}) \quad (1)$$

$$\lambda = N \times \Delta S, \quad (2)$$

где ΔS обозначает адаптивную длину дуги между точками, α - угловое разрешение, обеспечиваемое спецификацией Lidar, N - расчетный масштабный коэффициент относительного уровня шума Lidar, а λ - порог кластеризации. Следовательно, если расстояние между двумя точками скана

при одном и том же сканировании больше, чем λ , эти точки будут разделены на два разных кластера; то есть

$$\|p_i - p_{i-1}\| > \lambda \quad (3)$$

На основе ARC, сканирование, показанное на рисунке 4.1б, может быть разделено в две группы. Либо, сканы можно разделить на несколько сегментов из-за изменений радиуса, как показано на рисунке 4.1а. Таким образом, алгоритм ARC может разделять кластеры в соответствии с характеристиками измерения Lidar.

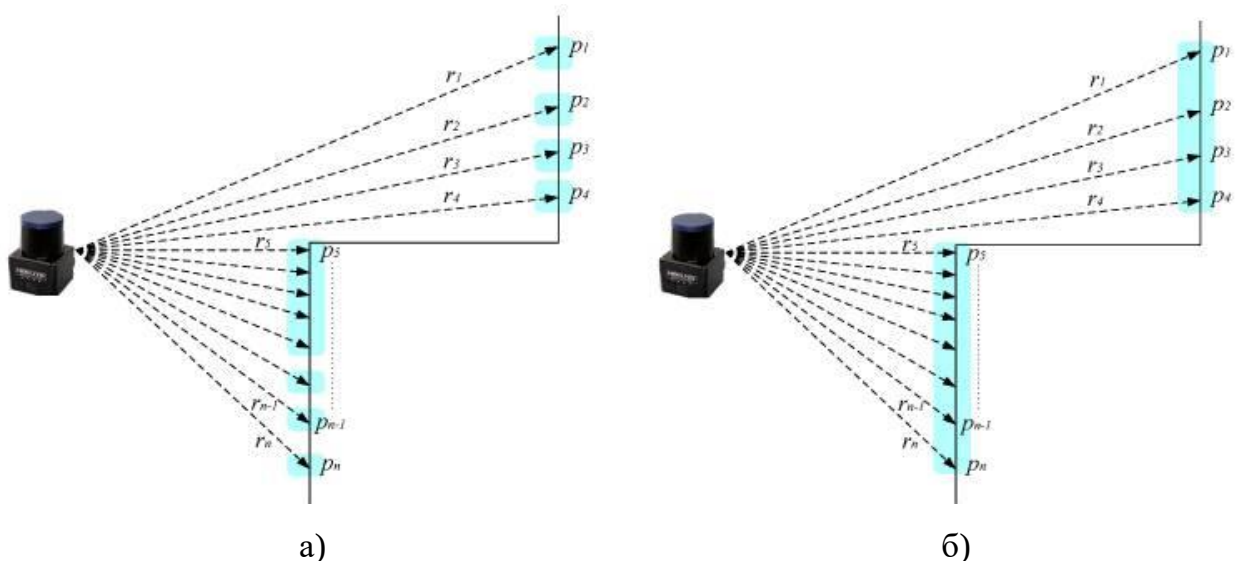


Рисунок 4.1 – Работа алгоритма адаптивной радиус-кластеризации: а) разделение на несколько сегментов из-за изменений радиуса; б) разделение на несколько групп из-за изменений радиуса

После использования алгоритма ARC кластеры с меньшим количеством точек сканирования рассматриваются как выбросы. Кластеры с более низкой плотностью точек, которые не могут предоставить достаточное количество информации, отбрасываются до этапа извлечения признаков[56].

4.1.3 Метод разделение и слияния для извлечения углов и линий

Сравнивая несколько методов извлечения признаков в различных аспектах, включая скорость алгоритма, точность, метод разделения и слияния рассматривается для извлечения признаков в этом исследовании. Алгоритм способен извлекать угловые характерные точки в среде, которые затем можно принять за стабильные и надежные характерные точки.

Процедура расщепления также объединяет идею итеративной подгонки конечной точки (IEPF), которая является рекурсивным алгоритмом для извлечения признаков. Для кластера $\Omega = \{x_i, y_i \mid i = 1, \dots, k\}$ алгоритм разделения и слияния соединяет первую точку и последнюю точку для образования прямой линии. Затем алгоритм вычисляет отклонения от всех точек этой линии. Если точка, в которой соответствующее максимальное расстояние больше предварительно определенного порога отклонения d_c , эта точка помечается как характерная точка, и она дополнительно разбивает кластер на два подмножества.

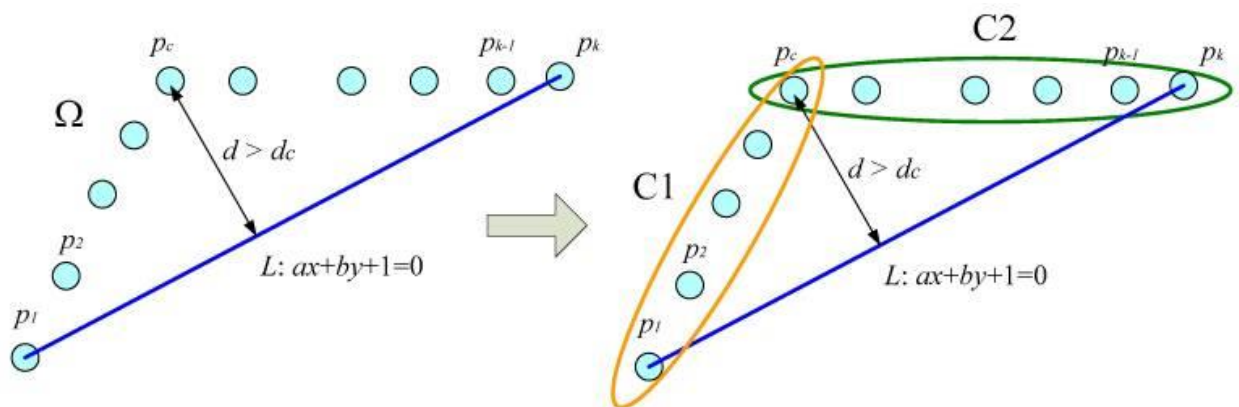


Рисунок 4.2 – Работа алгоритма разделения и слияния: а) процесс разбиения;
б) процесс слияния

На рисунке 4.2 показан основной процесс алгоритма разделения и слияния. Алгоритм рекурсивно разбивает множество точек на два подмножества, пока определенное заданное условие не будет выполнено.

4.2 Метод WP-ICP

4.2.1 Взвешенный параллельный алгоритм оценки положения ICP

Среда с подобным расположением обычно дает хорошие оценки. Однако на практике это не всегда так. Чтобы проверить осуществимость предложенного метода, предложенный алгоритм должен быть достаточно устойчивым даже при наличии неопределенностей окружающей среды, таких как движущиеся люди.

WP-ICP рассматривает две функции для облаков точек, которые предварительно обрабатываются: одна - угловая, а другая - линейная. Углы являются важными характерными точками в окружающей среде, поскольку они различны, в то время как стены являются устойчивыми характерными точками и хорошим кандидатом для извлечения элементов в структурированных средах. Используя преимущества Lidar для обнаружения препятствий, стены могут быть представлены в виде отрезков, состоящих из двух углов. Кроме того, возможно рассматривать центральную точку отрезка как другую подходящую контрольную точку.

Преимущество таких функций заключается в том, что внутренняя среда, например, офисы и здания, как правило, хорошо структурирована. Поэтому для таких сред подходит локализация на основе функций. Изучая традиционный алгоритм ICP для полных точек, регистрация облака точек достигается с помощью концепции поиска ближайших соседей (NNS). Там нет дополнительной информации, прикрепленной к данным точкам. Таким образом, легко получить неправильное соответствие, как показано на рисунке 4.3. В этом случае, обычно, требуется несколько итераций для

схождения регистрации точки. Кроме того, ICP приводит к очевидным затратам времени на регистрацию, особенно когда размер облака точек велик. Чтобы решить проблемы с неправильной корреспонденцией и стоимостью итерации, разработан метод уменьшения облаков точек на основе признаков.

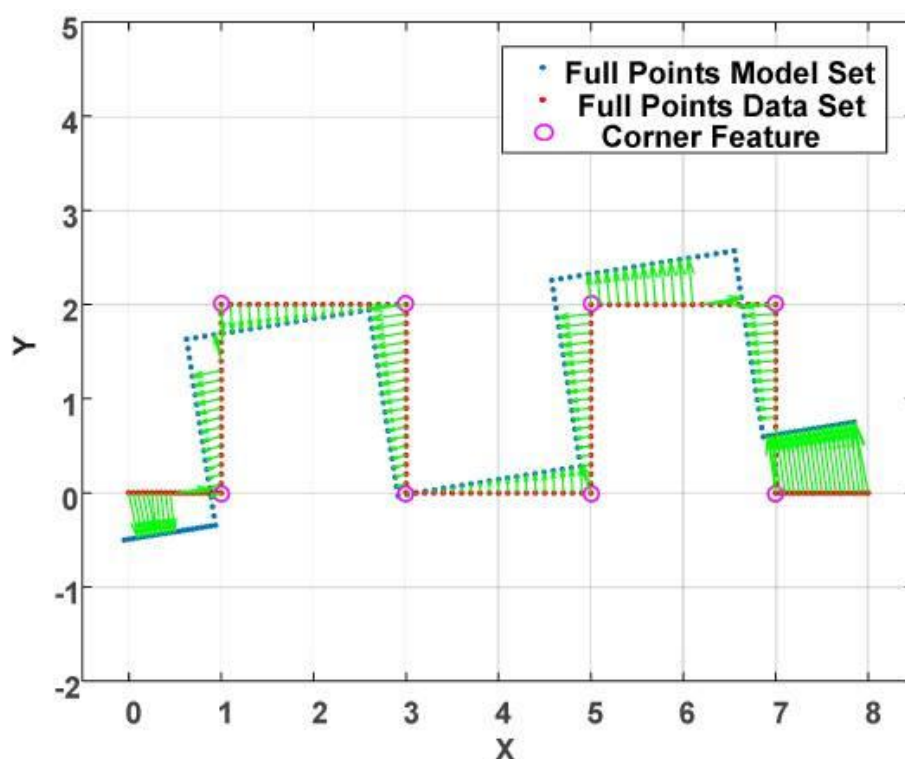


Рисунок 4.3 – Ложные регистрации точек, вызванные алгоритмом итеративного поиска ближайшего соседа (NNS)

Для рассматриваемого метода WP-ICP облако точек сначала характеризуется меньшим количеством углов и линий. Это имеет два основных преимущества.

Во-первых, размер облака точек значительно уменьшается и, таким образом, увеличивает скорость ICP.

Во-вторых, отполированные точки помечены как угловые или линейные элементы. Только те точки, которые помечены как углы, могут быть сопоставлены с кандидатами на углу; аналогично, только те точки,

которые помечены как линии, могут быть сопоставлены с линиями-кандидатами.

Результат показан на рисунке 4.4. Рисунок 4.4а иллюстрирует условие соответствия для углов, а рисунок 4.4б представляет условие соответствия для линий. Параллельный ICP приводит к правильной регистрации точки и тем самым уменьшает количество итераций.

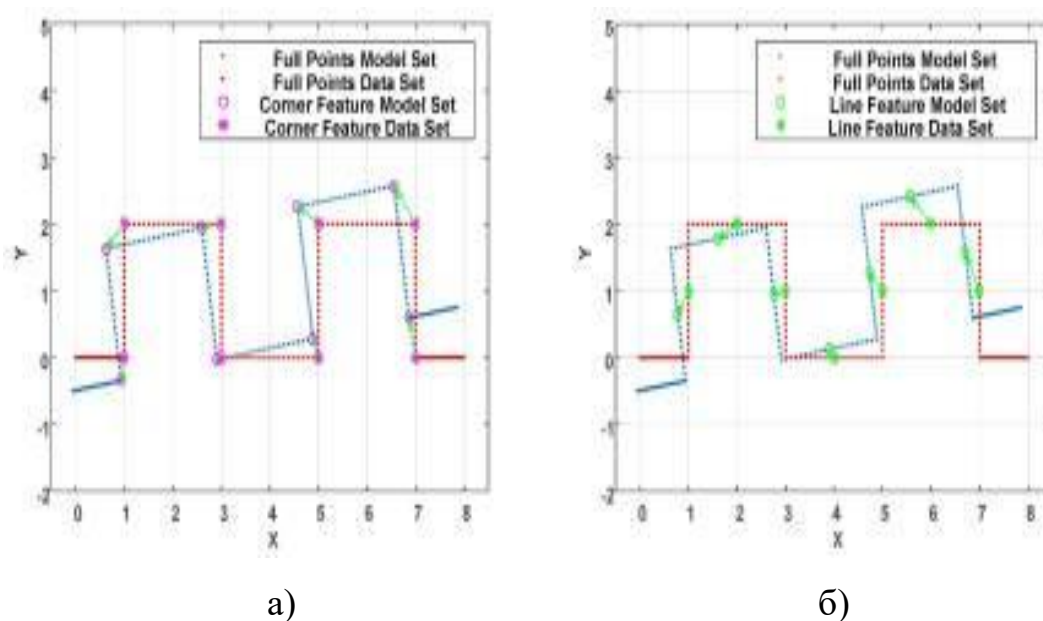


Рисунок 4.4 – Регистрации облака точек на основе объектов: а) угловые элементы сопоставлены с соответствующими угловыми элементами; б) линейные элементы сопоставлены с соответствующими линейными элементами

Основное преимущество WP-ICP заключается в том, что количество точек данных в наборе данных, а также в наборе моделей может быть значительно уменьшено. Напротив, алгоритм ICP с полными точками включает в себя все точки данных для поиска соответствия и, таким образом, приводит к низкой эффективности вычислений.

Кроме того, в WP-ICP точки сканирования сгруппированы в угловые или линейные группы, соответственно. При использовании метода WP-ICP,

основанного на параллельном механизме, точки из набора углов никогда не могут быть сопоставлены с точками из набора линий. Таким образом, можно избежать несоответствия при регистрации точек. Однако в ICP может произойти много несоответствий, если расстояния между этими двумя заданными точками достаточно близки.

Так как WP-ICP имеет два процесса ICP одновременно, он генерирует две пары положения робота, а именно (RC, tC) и (RL, tL). Поэтому желательно объединить эти две положения, для получения более достоверной информации о текущем положении.

Поскольку WP-ICP предоставляет два источника характерных точек, точки сканирования Lidar в реальном времени будут разделены на две группы, включая углы и линии. Затем каждая группа сопоставляется со своими соответствующими характеристиками. На основе этого механизма параллельного сопоставления можно избежать серьезного несоответствия, что приведет к повышению стабильности и точности локализации.

Практические преимущества рассматриваемого WP-ICP включают в себя следующее:

- 1) вычислительные затраты уменьшаются, когда создаются деревья KD и применяется регистрация ближайшей точки;
- 2) правильная регистрация облака точек значительно сокращает количество итераций ICP, что позволяет быстро оценить позу робота;
- 3) положение робота определяется путем объединения двух характеристик (угла и линии) на основе признаков ICP, что делает оценку менее чувствительной к неопределенным условиям.

5 Алгоритмы программного комплекса пространственной навигации и мониторинга на основе визуальной одометрии

Общая структура алгоритмов программного комплекса на основе визуальной одометрии представлена на рисунке 5.1. В целом под позицией камеры понимается как положение, так и ориентация, то есть полное преобразование 6 степеней свободы. Ориентировочная система координат окружающего пространства устанавливается в позиции первого кадра всей последовательности.

5.1 Описание работы системы визуальной одометрии

Первым шагом является обработка системой первого полученного кадра, полученного монокулярного датчика, стереометрического датчика, либо от датчика RGB-D. В случае стереокамеры, извлеченный стереометрический кадр состоит из синхронизированных изображений с левой и правой камер. Стереокادر должен пройти процедуру стерео выпрямления. Стерео выпрямление - это процесс виртуальной трансформации стереокадра, так что создается впечатление, что две камеры стереосистемы имеют выравнивание плоскостей изображения, чтобы они были компланарными. Следовательно, эпиполярные линии теперь параллельны базовой линии стерео между камерами. Это сводит проблему стерео соответствия к одномерному поиску, поскольку совпадающие элементы между двумя камерами будут лежать в одном ряду пикселей, предполагая горизонтальную стереосистему. В случае датчика RGB-D, изображение RGB преобразуется в изображение в градациях серого, а затем подается вместе с изображением глубины в систему.

В качестве метода, основанного на элементах, характерные точки или угловые элементы будут извлекаться и использоваться для последующей

обработки. Угловые функции вычисляются достаточно быстро, так как, на текущий момент, доступно много различных угловых детекторов. В этой работе используется адаптивный и общий ускоренный сегментный угловой детектор (AGAST). AGAST основан на функциях углового детектора ускоренного тестирования сегментов (FAST), который обладает высокой степенью сходимости и ускоренными вычислительными характеристиками. AGAST улучшает ускоренный сегментный тест, который лежит в основе FAST, делая его более универсальным и повышая его производительность. В данном случае не строится масштабная пирамида изображения, а углы извлекаются из полноразмерных изображений.

Для каждой обнаруженной функции вычисляется дескриптор функции. Дескриптор функции действует как подпись для этой функции. Сопоставление между функциями может быть выполнено путем сравнения их дескрипторов. В этой работе используются дескрипторы двоичных робастных независимых элементарных признаков (BRIEF).

КРАТКИЕ дескрипторы - это дескрипторы двоичных объектов, то есть дескрипторы в форме двоичной строки. Это означает, что сопоставление происходит быстрее, в зависимости от расстояния Хэмминга. Расстояние Хэмминга определяется как количество позиций, в которых соответствующие биты различны. Расстояние Хемминга вычисляется достаточно просто и быстро вследствие того, что это просто операция с исключающими ИЛИ (XOR) и счетчиком битов, то есть сумма ($xor(descriptor1, descriptor2)$). Однако КРАТКИЕ дескрипторы не имеют инвариантности вращения.

Важной проблемой для рассмотрения является распределение обнаруженных признаков по изображению. Плохое распределение – распределение, когда обнаруженные объекты, сосредоточенные в одной области изображения, приводят к искажению результатов. Необходимо прибегнуть к двухэтапному процессу, чтобы преодолеть эту проблему.

Сначала изображение делится на ячейки, где функции будут обнаруживаться в каждой ячейке отдельно. Затем, метод, известный как адаптивное не-максимальное подавление, выполняется в каждой ячейке. Адаптивное не-максимальное подавление направлено на ограничение максимального количества извлекаемых объектов, в то же время, обеспечивая хорошее распределение по изображению. Элементы подавляются на основании веса углов, и сохраняются только те из них, которые являются локальными максимумами по соседству.

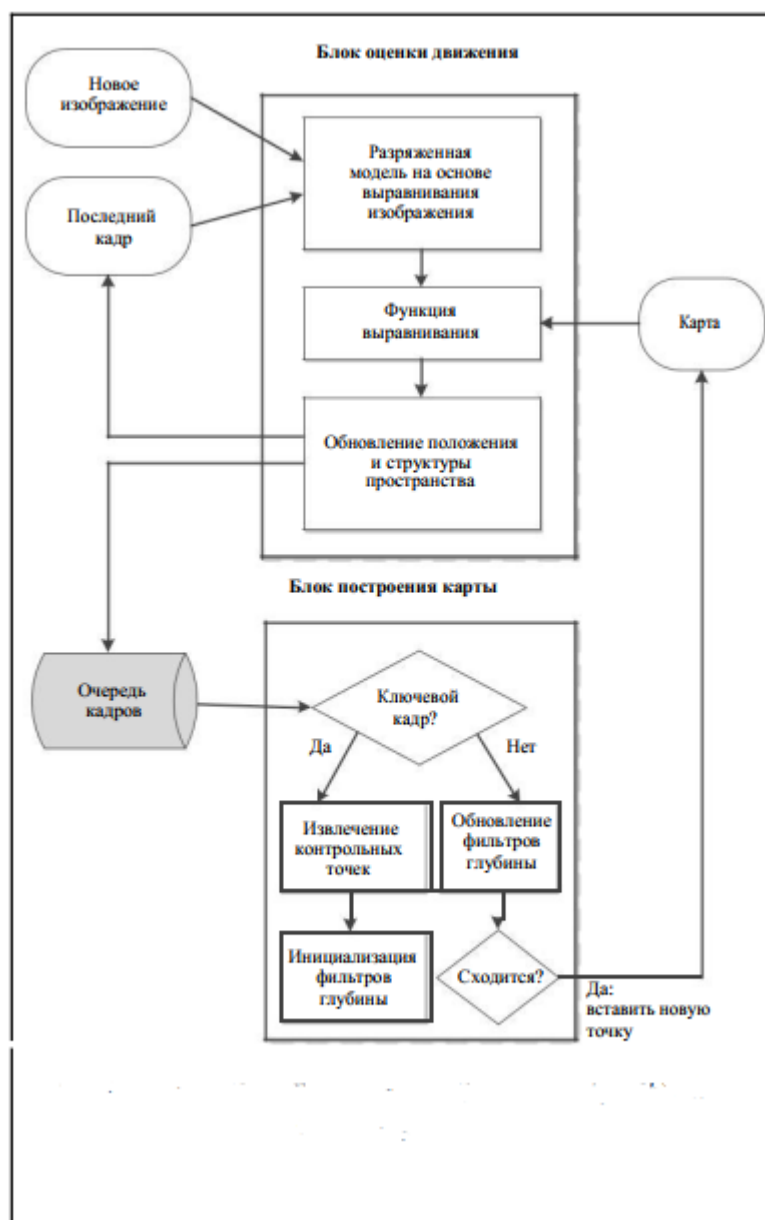


Рисунок 5.1 – Структура алгоритмов программного комплекса на основе визуальной одометрии

6 Описание концепции и инструментов фреймворка ROS

6.1 Общие сведения

Robot Operating System (ROS) — это широко используемый в робототехнике фреймворк. Философия ROS заключается в создании программного обеспечения, которое позволяет вам работать с различными роботами, только внося небольшие изменения в код. Эта идея позволяет создавать функциональные возможности, которые могут быть просто перенесены для использования различными роботами.

ROS имеет стандартные функции операционной системы, такие как аппаратная абстракция, низкоуровневое управление устройствами, часто используемые функции, межпроцессная передача сообщений и управление библиотеками. Архитектура ROS не основана на графе с централизованной топологией. Обработка происходит в узлах, которые могут получать или отправлять данные от датчиков, систем контроля состояния и состояния накопителей. Библиотека ориентирована на Unix-подобные системы.

Пакет `*-ros-pkg` - это общий репозиторий для разработки высокоуровневых библиотек. Многие функции, часто связанные с ROS, такие как библиотеки навигации и визуализатор RViz, хранятся в этом хранилище. Эти библиотеки предоставляют мощный набор инструментов (различные визуализаторы, симуляторы, инструменты отладки) для упрощения работы.

ROS распространяется на условиях лицензии BSD и является программным обеспечением с открытым исходным кодом. ROS бесплатен для исследовательских и коммерческих целей.

ROS поддерживает параллельные вычисления, имеет хорошую интеграцию с популярными C++ библиотеками, такими как *OpenCV*, *Qt*, *Point Cloud Library* и пр., и она может работать на одноплатных

компьютерах, таких как *Raspberry Pi* или *BeagleBone Black*, а также с микроконтроллерными платформами, например, *Arduino* [57].

6.2 Архитектура ROS

В архитектуре ROS можно выделить три концептуальных уровня:

- 1) уровень файловой системы (*Filesystem level*);
- 2) уровень вычислительного графа (*Computation Graph level*);
- 3) уровень сообщества (*Community level*).

Первый уровень — это уровень файловой системы. На этом уровне расположена внутренняя структура ROS — структура папок, файлы, необходимые для работы.

Второй уровень — это уровень вычислительного графа, на котором происходит взаимодействие между процессами и системами. На этом уровне находятся концепции и модули, которые имеются в ROS для создания систем, обработки всех процессов, коммуникации с более чем одним компьютером и так далее.

Третий уровень — это уровень сообщества. Этот уровень содержит инструменты и концепции для обмена знаниями, алгоритмы и код от любого разработчика.

6.3 Файловая система ROS

Первым уровнем в архитектуре ROS является уровень файловой системы.



Рисунок 6.1 – Файловая система ROS

Как и в случае обычной операционной системы, программы в ROS разделены на папки, в которых содержатся некоторые файлы, описывающие ее функциональность:

1) пакеты: реализуют базовый уровень ROS. Пакет имеет минимальную структуру и содержимое, чтобы создать программу в ROS. Он может иметь выполняемые процессы (узлы или *node*), файлы конфигурации и так далее;

2) декларации: содержится информация о пакетах, лицензионная информация, зависимости, флаги компиляции и прочее. Управление декларациями осуществляется через файл `manifests.xml`;

3) стеки: при объединении вместе нескольких пакетов для, получения некоторой функциональности, формируется так называемый стек. В ROS, реализована возможность создания и множественной конфигурации таких стеков для различных целей, например, стек навигации;

4) декларации стеков: предоставляют данные о стеке, включая его структурную информацию и его зависимости от других стеков;

5) типы сообщений: сообщение является информацией, которую процесс отправляет другим процессам. В ROS имеется множество стандартных типов сообщений.

б) типы сервисов (Service types, src): служат для определения в ROS структуры данных запросов и ответов для сервисов.

6.4 Уровень вычислительного графа

ROS создает сеть, в которой соединены все процессы. Любой узел в системе может получить доступ к этой сети, взаимодействовать с другими узлами, смотреть информацию, которую они посылают, и передавать данные в сеть.

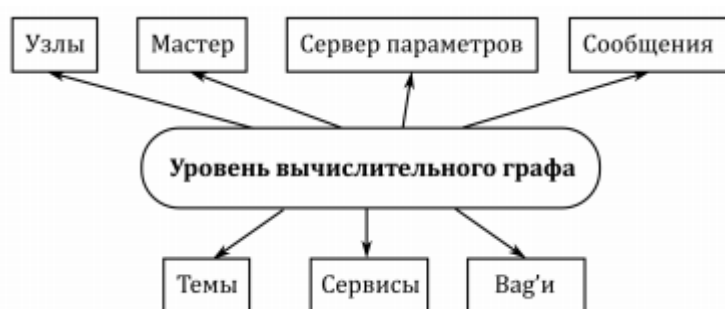


Рисунок 6.2 – Уровень вычислительного графа ROS

Базовыми понятиями на этом уровне являются: узлы, мастер, сервер параметров, сообщения, сервисы, темы и бэги. Все они различными способами обеспечивают граф данными [57]:

1) узлы: это вычислительные процессы. Если вам нужен процесс, который может взаимодействовать с другими узлами, вам необходимо создать узел с этим процессом, подключив его к сети ROS. Обычно система имеет много узлов для управления различными функциями. Узлы записываются в клиентскую библиотеку ROS;

2) мастер: обеспечивает регистрацию имени и поиск оставшихся узлов. Если его нет в системе, вы не сможете обмениваться информацией с узлами, сервисами, сообщениями и другими потоками. Мастер может находиться на компьютере, где узлы работают с другими компьютерами;

3) сервер параметров: позволяет сохранять данные, используя ключи, хранящиеся централизованно. С помощью этого параметра можно настроить узлы во время их работы или изменить рабочий узел;

4) сообщения: узлы взаимодействуют друг с другом посредством сообщений. Сообщение содержит данные, передающие информацию другим узлам. ROS имеет много типов сообщений. Можно создать свой собственный тип сообщения, используя набор стандартных сообщений;

5) темы: каждое сообщение должно иметь имя для отправки в сеть ROS. Когда узел является отправителем данных, это означает, что узел опубликовал тему. Узлы могут получать темы с других узлов, просто подписавшись на тему. Узел может подписаться на тему, и необязательно, чтобы узел, который будет публиковать тему, существовал в данный момент. Важно, чтобы название темы было уникальным, чтобы избежать проблем с передачей данных между темами с одинаковыми именами;

6) сервисы: сервисы предоставляют возможность взаимодействия с узлами. Сервисы также должны иметь уникальное имя. Когда у некоторого узла есть сервис, все узлы могут связываться с ней благодаря клиентским библиотекам ROS;

7) бэги: формат для хранения и воспроизведения данных сообщения ROS. Пакеты являются важным механизмом хранения данных, таких как данные датчиков, которые сложно собрать, но они необходимы для разработки и тестирования алгоритмов. Бэги часто используются при работе со сложными роботами.

7 Разработка автоматизированной системы управления и ее функции

7.1 Назначение системы

Разрабатываемая САУ мобильной навигационной системы реализует следующие функции:

- 1) сбор, обработка и анализ информации о текущей ориентации платформы в пространстве на основе данных с визуальных датчиков и датчиков типа лидар;
- 2) осуществление дистанционного управления посредством пульта управления;
- 3) осуществление вычисления и анализа данных о навигации платформы относительно заданной точки на основе комплексной обработки информации, получаемой от визуальных датчиков и датчиков типа лидар;
- 4) реализацию и контроль выполнения управляющих воздействий.

7.2 Описание системы

Проектируемая АС имеет трехуровневую автоматизированную систему.

Нижний уровень состоит из источников данных:

Задачи устройств данного уровня:

- 1) сбор информации об ускорении, угловой скорости для определения текущей ориентации технологического объекта в пространстве;
- 2) передача и прием сигнала для осуществления дистанционного управления.

Средний уровень состоит из платформы и датчика типа лидар. Навигационная платформа выполняет фактически все вычислительные операции в системе:

- 1) обработка и преобразование данных, поступающих с низкого уровня;

2) выполняет алгоритм навигации платформы, обеспечивает адекватную реакцию на дистанционное управление;

Верхний уровень образует автоматизированное рабочее место (АРМ) оператора. Управление платформой осуществляется в полуавтоматическом режиме по командам оператора.

7.3 Описание подсистем

Данная система с точки зрения программного обеспечения подразделяется на такие подсистемы, как:

- 1) подсистема сбора данных;
- 2) подсистема калибровки и обработки данных;
- 3) подсистема определения текущей ориентации объекта в пространстве;
- 4) подсистема дистанционной связи;
- 5) подсистема выработки управляющих воздействий на двигатели.

8 Синтез модели мобильного робота на основе алгоритмов SLAM-навигации

8.1 Выбор структуры автоматической системы регулирования

Робототехническая платформа состоит из платформы, на которой находится вся бортовая аппаратура, приводящая в движение робота. Движение выполняется с помощью 2 двигателей, воздействующих на одно из двух ведущих колес робота. Двигатели запускаются от сигналов, подаваемых с драйвера двигателей постоянного тока, которые питаются от аккумулятора. Основным сигналом для пускателя является сигнал, подаваемый с микропроцессорного контроллера, который, на основе преобразованной информации с датчика типа лидар, вырабатывает команды, воздействующие на исполнительные механизмы в соответствии с алгоритмом. В роли визуального датчика могут выступать камеры видимого или инфракрасного диапазона, стереокамеры и камеры глубины. С помощью персонального компьютера и программного обеспечения изображение, полученное камерами, преобразуется в изменение координат по ширине и длине.

В качестве метода измерения местоположения выбираем метод SLAM (Simultaneous Localization and Mapping) – метод одновременной локализации и построения карты.

Основная идея алгоритма SLAM заключается в том, что, находясь в некотором положении, робот начинает строить карту окружающего пространства. Сделав первые замеры расстояний, он их запоминает и движется в направлении других объектов. После того, как карта считается достроенной, в некотором помещении найдены его особенности и расстояние до них измерено робот возвращается на исходную позицию. Скорее всего, он попадёт в другое место, а не в то, в котором находился изначально. Происходит это из-за ошибок одометрии. Далее робот начинает второй круг измерений, на котором ошибка сокращается.

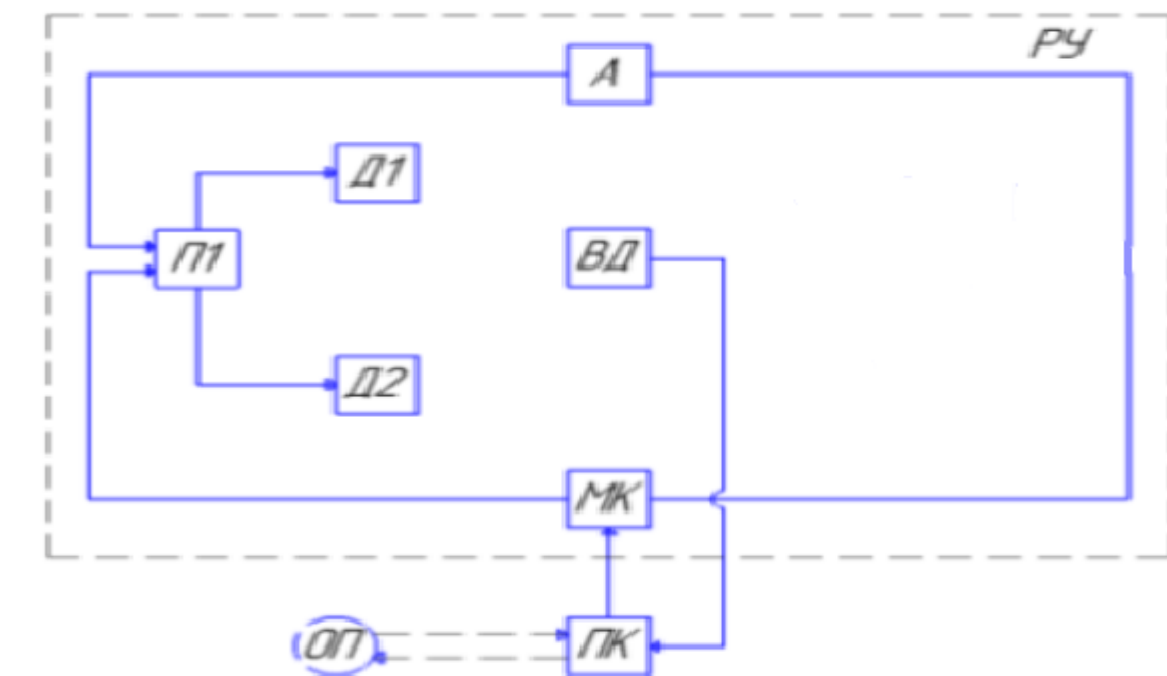


Рисунок 8.1 – Структурная схема РУ:

Д1, Д2 – первый и второй двигатели; А – аккумулятор; П1– драйвер двигателя; ВД – визуальный датчик; МК – микропроцессорный контроллер; ПК – персональный компьютер; ОП – оперативный персонал; РУ – робототехническое устройство.

Описание функционирования АСР:

- 1) сигнал с ВД поступает на ПК, где обрабатывается, согласно алгоритму и сохраняется в виде матрицы расстояний до точек окружающего пространства;
- 2) преобразованный сигнал поступает на МК, где вырабатывается команда по управлению исполнительными механизмами в соответствии с одним из законов регулирования;
- 3) пускатель П1, питаясь от аккумулятора, включает или выключает двигатели по команде от МК;
- 4) двигатели приводят в движение РУ, изменяя его координаты;
- 5) пункты 1-4 повторяются до тех пор, пока робот, пройдя всю заданную местность, не вернется в начальную позицию.

8.2 Разработка структурной схемы подсистемы навигации

Перед настройкой среды, в которой будет вестись разработка, очень важно знать ограничения стека навигации для дальнейшей адаптации оборудования. Существует четыре основных ограничения в отношении аппаратного обеспечения:

1) стек был создан с целью охвата только дифференциального привода и голономных роботов, хотя можно использовать некоторые функции с другими типами роботов, которые здесь не рассматриваются;

2) навигационный стек предполагает, что робот получает сообщение типа *twist_message* со скоростями X, Y и Theta и может управлять мобильной базой для достижения этих скоростей. Если интерфейс робота не в состоянии это сделать, требуется адаптировать оборудование или просто создать узел *ros*, который преобразует *twist_message*, предоставляемое стеком навигации, в тип сообщения, которое лучше всего соответствует интерфейсам робота;

3) информация об окружающей среде собирается из темы типа сообщений *LaserScan*. Для плоских лазеров, таких как Hokuyo URG или SICK Laser, необходимо публиковать свои данные, для чего, достаточно установить узел *hokuyo*, *sicktoolbox* или аналогичные пакеты, в зависимости от типа датчика. Кроме того, можно использовать другие датчики, если имеется возможность преобразовать их данные в тип *LaserScan*;

4) навигационный стек будет работать лучше с квадратными или круглыми роботами, тогда как его можно использовать с произвольными формами и размерами. Уникальные размеры и формы могут вызвать проблемы с роботом в ограниченном пространстве.

Только при соблюдении всех данных условий для использования стека навигации выполнены, можно гарантировать его стабильную работу. Поскольку обзор структуры автоматической системы регулирования уже был сделан в разделе 8.1, здесь рассматривается только структура подсистемы

навигации программного навигационного стека, который представлен на рисунке 8.2.

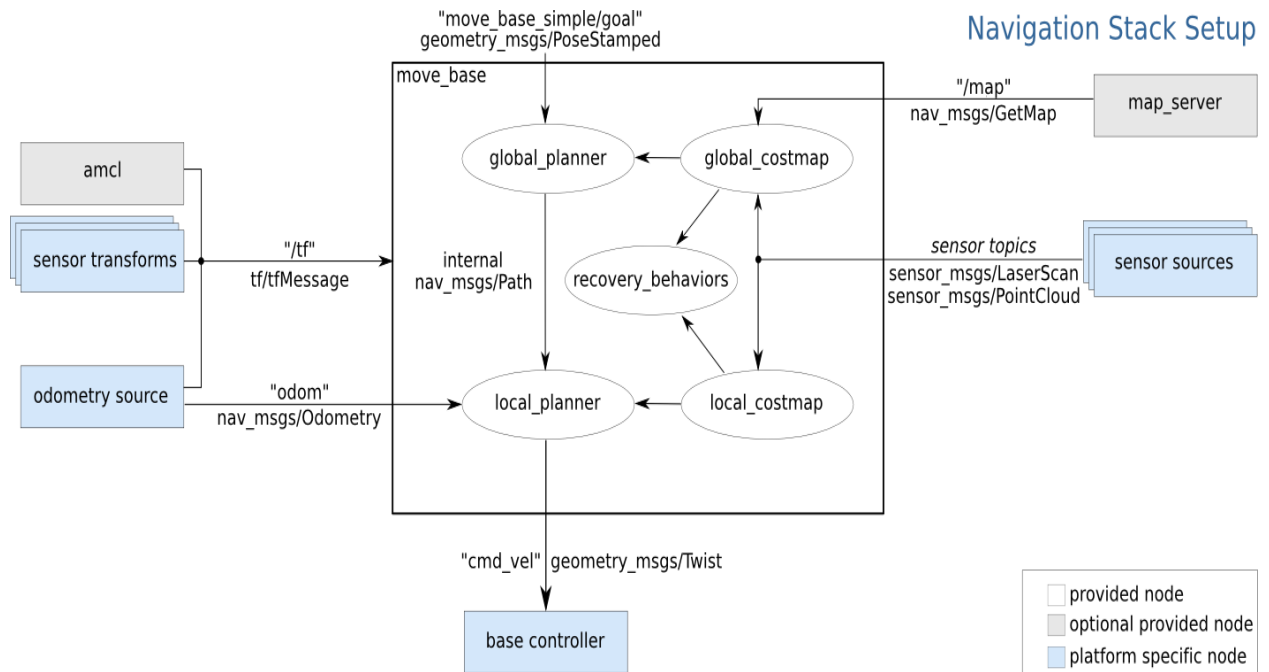


Рисунок 8.2 – Структурная схема подсистемы навигации

AMCL и картографический сервер. Данные два блока, отвечают за использование статической карты. Картографический сервер содержит два узла: сам картографический сервер и вспомогательные модули, направленные на упрощение работы с ним, а также на улучшение стабильности его работы. Первый пакет является узлом ROS, который предоставляет статические данные карты в качестве службы ROS, а группа вторых выполняет функцию хранителя карты, сохраняя динамически сгенерированную карту в файл. AMCL не обладает функционалом для управления картами, это фактически система локализации, работающая на известной карте. Эта система локализации основана на подходе локализации Монте-Карло: она случайным образом распределяет частицы в известной карте, представляющие возможные местоположения робота, а затем использует фильтр частиц для определения фактической позы робота.

Google Cartographer. Google Cartographer, является системой локализации, но, в отличие от `amcl`, он работает в неизвестной среде, выполняя одновременную локализацию и картографирование (SLAM). Он создает двухмерную карту сетки занятости с использованием данных о положении робота и лазерных данных (или преобразованных данных, таких как данные сенсора Kinect).

Датчики и контроллер. Эти блоки системы относятся к аппаратно-программному взаимодействию и, как указано, являются узлами, специфичными для платформы. Источник одометрии и блоки базового контроллера относятся к используемому роботу, поскольку первый обычно публикуется с использованием данных колесных энкодеров, а второй отвечает за получение данных о скорости из темы `cmd_vel` и гарантирует, что движение робота соответствует направлению вектора данной скорости.

Локальные и глобальные карты затрат. Локальные и глобальные 2D карты затрат - это темы, содержащие информацию, которая представляет проекцию препятствий в 2D плоскости (пол), а также радиус инфляции безопасности, область вокруг препятствий, которые гарантируют, что робот не будет сталкиваться с какими-либо объектами, независимо от того, какова его ориентация. В то время как глобальная карта затрат представляет всю среду (или огромную ее часть), локальная карта затрат, как правило, представляет собой окно прокрутки, которое перемещается в глобальной карте затрат относительно текущей позиции робота.

Локальные и глобальные планировщики. Локальные и глобальные планировщики не работают одинаково. Глобальный планировщик берет текущую позицию робота и цель и отслеживает траекторию более низкой стоимости относительно глобальной карты затрат. Однако у локального планировщика есть более интересная задача: он работает над локальной картой затрат, и, поскольку локальная карта затрат меньше, она обычно имеет большее определение и, следовательно, способна обнаруживать

больше препятствий, чем глобальная карта затрат. Таким образом, местный планировщик отвечает за создание развертывания траектории по глобальной траектории, которая способна вернуться к исходной траектории с меньшими затратами. Блок управления движением мобильного робота *move_base* - это пакет, который содержит в себе локальные и глобальные планировщики и отвечает за их связь для достижения цели навигации.

8.3 Разработка подсистемы телеуправления

Главным элементом подсистемы управления является вектор управления. Вектор управления предназначен для передачи команд с подсистемы высокого уровня на подсистему низкого уровня. Команды формируются в автоматическом режиме на основе результатов вычислений алгоритма SLAM-навигации, либо в режиме ручного управления, когда команды для движения платформы формируются непосредственно оператором. Ниже представлена функциональная блок схема алгоритма работы вектора управления.

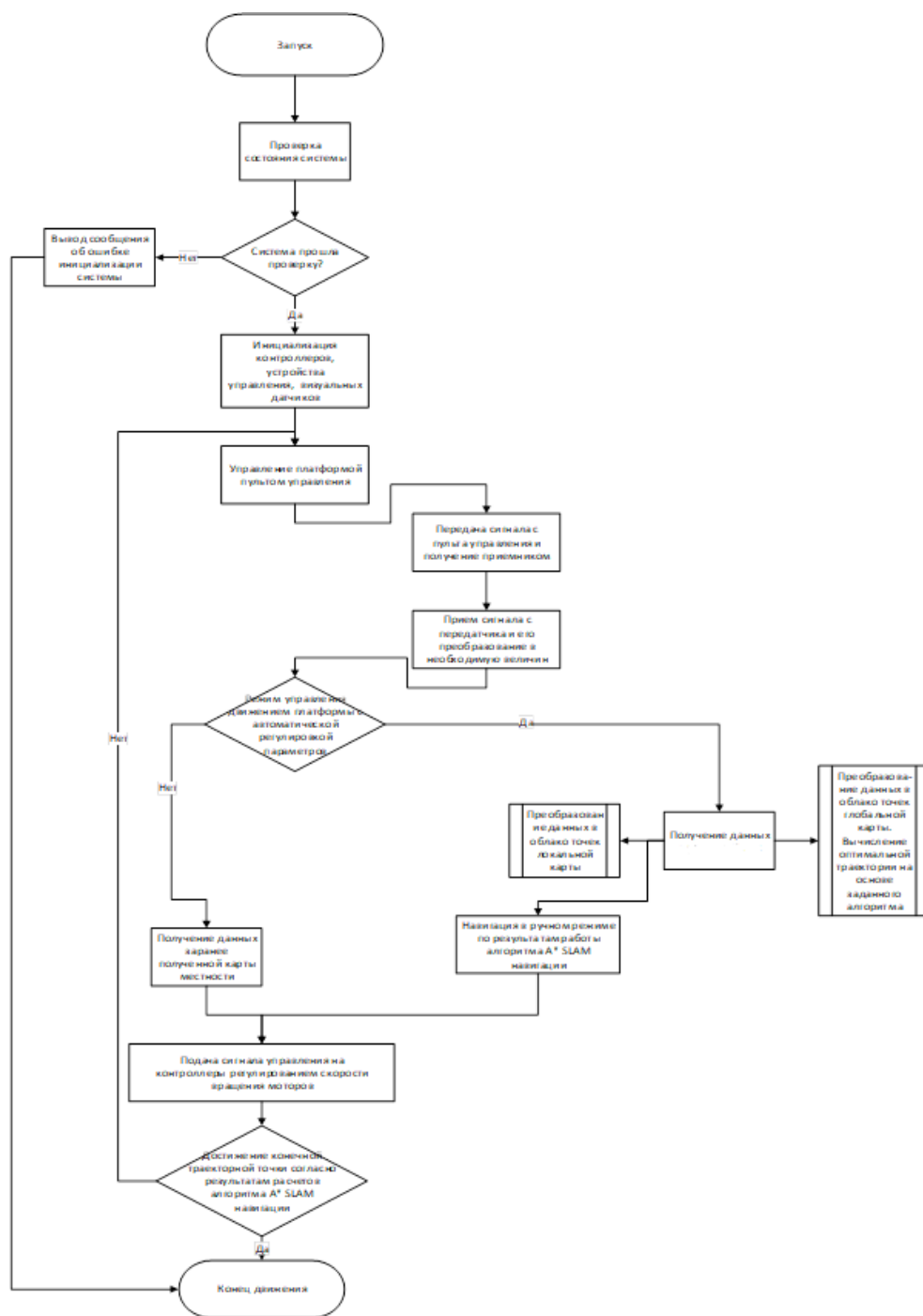


Рисунок 8.3 – Функциональная схема блока вектора управления подсистемы навигации

В качестве входных параметров для анализа принимаются данные о состоянии глобальной и локальной карты, полученных в программном пакете RTabMap_ROS на основе датчика визуальной информации.

Выходными параметрами являются векторы линейной и угловой скорости, которые и являются фактическими командами для осуществления передвижения платформы.

8.4 Сравнение работы методов VSLAM по качеству визуальной одометрии

В этом разделе приведены результаты работы монокулярных систем SLAM на базе ROS.

Все системы были протестированы на данных с монокулярной камеры Logitech с частотой кадров около 20 кадров в секунду. Обработка систем SLAM обеспечивалась внешним ПК. Основным недостатком этих систем является проблема неоднозначности масштаба, что означает отсутствие информации об окружающей среде в абсолютном масштабе. Мы используем ручную обработку для решения этой проблемы.

Параллельное отслеживание и картирование (PTAM). PTAM является одной из основанных на ключевых кадрах визуальных систем SLAM, которая позволяет оценивать трехмерную позу робота и строить трехмерную карту, отслеживая функции FAST. Система получает данные с датчика *msgs / Image* и публикует геометрию *msgs / PoseWithCovarianceStamped*, *tf* преобразование. Система не выводит карту в среду ROS, поэтому карту можно наблюдать с помощью встроенного инструмента. Нам не удалось получить надежные данные из этой системы после двух кругов UGV. Система потеряла след, когда робот повернул, демонстрируя недостаточную устойчивость к боковым маневрам.

Крупномасштабный прямой монокулярный SLAM (LSD SLAM). LSD SLAM - это прямой метод, который использует локальное изображение для подгонки изображения, обеспечивая прогнозирование позы и построение плотной карты. Система LSD SLAM имеет опцию замыкания контура,

которая дает преимущество для компенсации ошибки дрейфа. Система имеет плохую интеграцию с ROS, подписывается на датчик msgs / Image и публикует трансформации. После масштабной коррекции траектория мобильного робота была восстановлена и сравнена с траекторией ORB-SLAM.

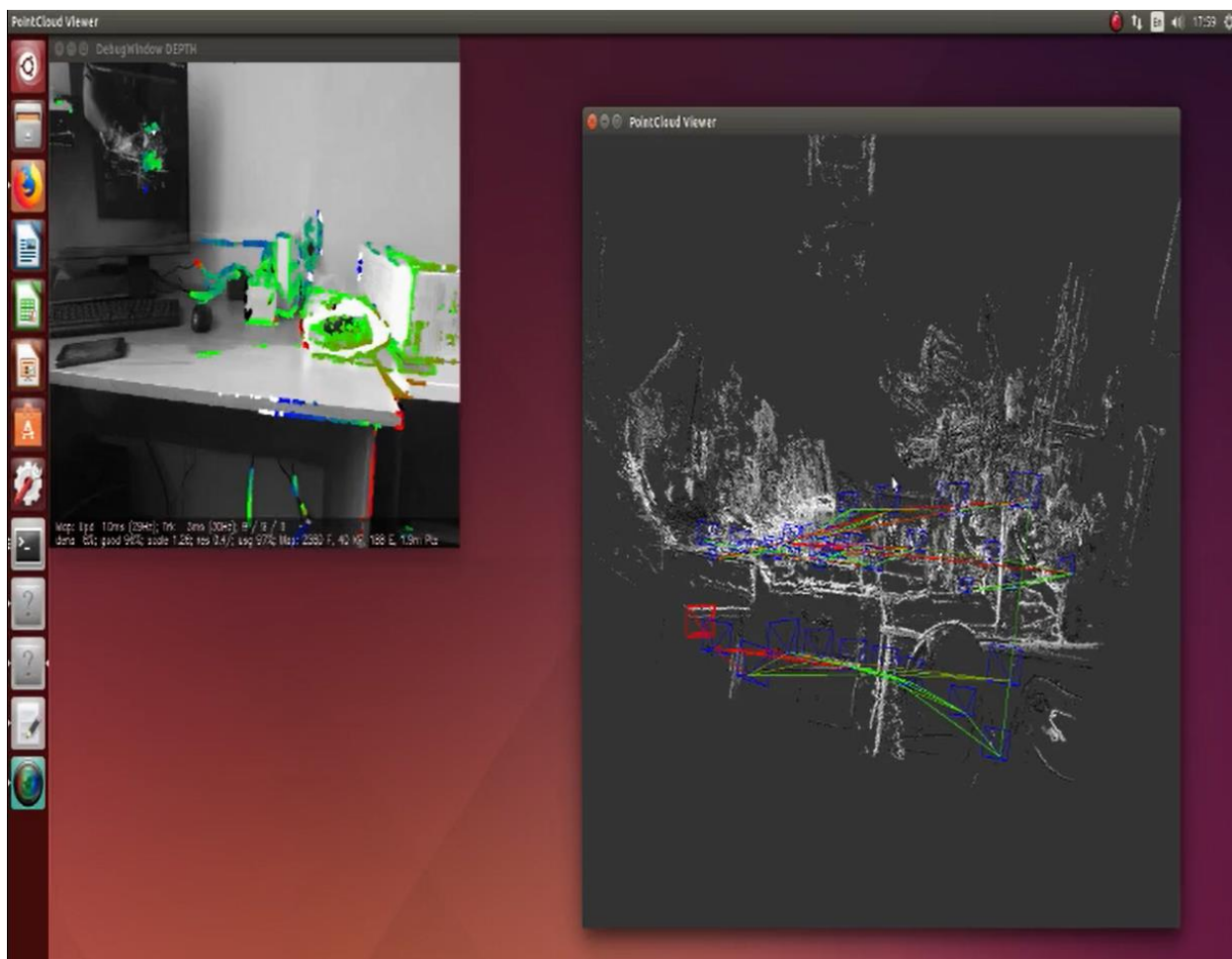


Рисунок 8.4 – Реализация работы LSD SLAM

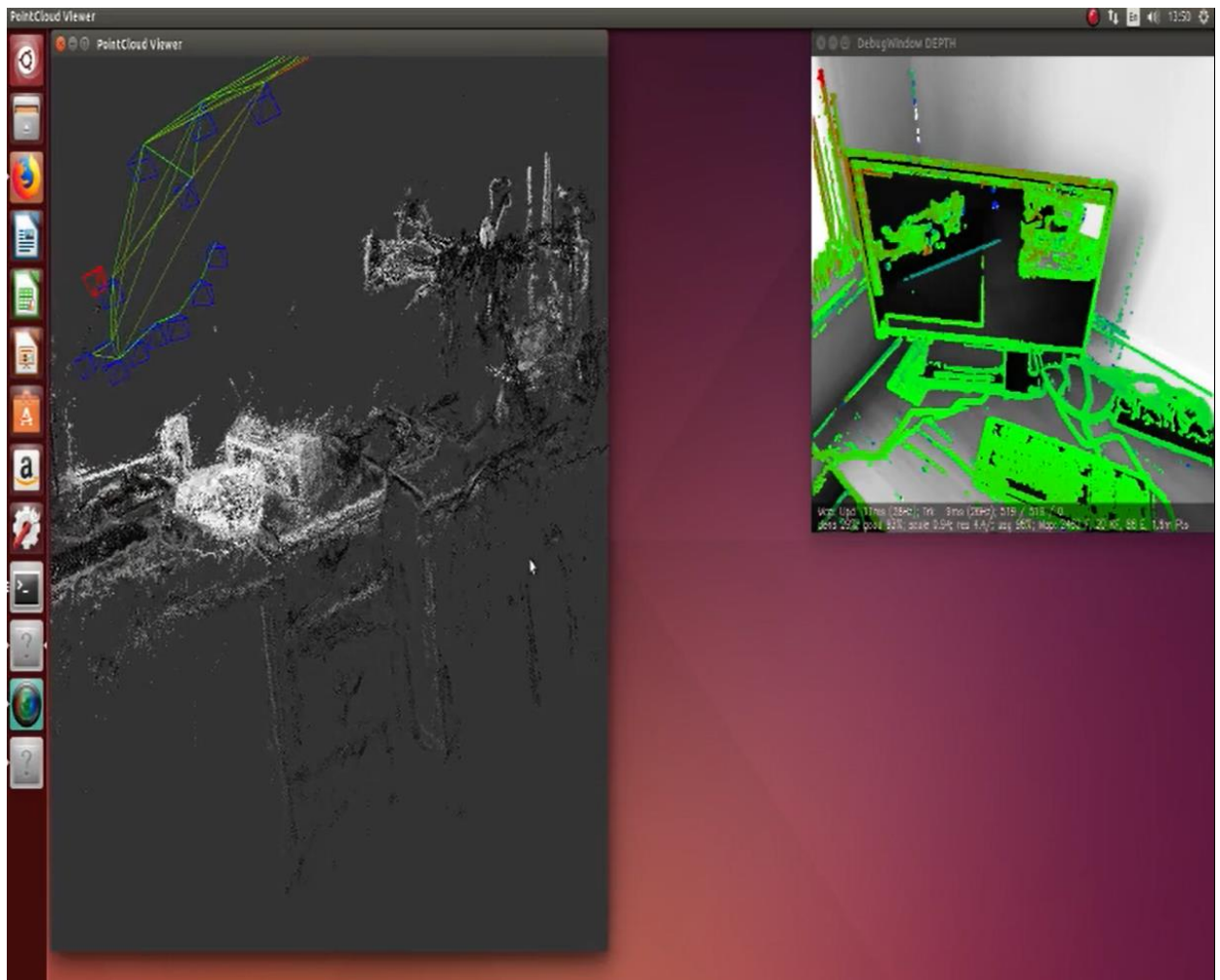


Рисунок 8.5 – Реализация работы LSD SLAM

Система имеет проблему при вычислении положения робота при смещении в начале построения карты, поскольку карта недостаточно плотна для точной локализации. Это смещение может быть отфильтровано с помощью уточнения модели движения робота. Таким образом, система является надежной и после абсолютного восстановления масштаба может использоваться для оценки позы робота и построения карты.

ORB SLAM. ORB SLAM - это система, основанная на функциях, которая отслеживает возможности ORB для оценки положения робота. Создает разреженное облако точек в качестве карты. Система имеет функцию обнаружения замыкания петли и широко используется для мобильных роботов, используемых в помещениях. Процесс обнаружения

особых точек и карта с оценкой положения представлены на рисунке 8.6. Система требует дополнительной разработки для интеграции в ROS. Траектория камеры может быть оценена только после обработки набора данных. Карта является хорошим приближением к тестовой среде. Оценка позы предоставляется только во время ключевых кадров, а информация о местоположении камеры более скудна. Система ORB SLAM очень надежна и обеспечивает хорошее приближение траектории робота, если решена задача восстановления в абсолютном масштабе.

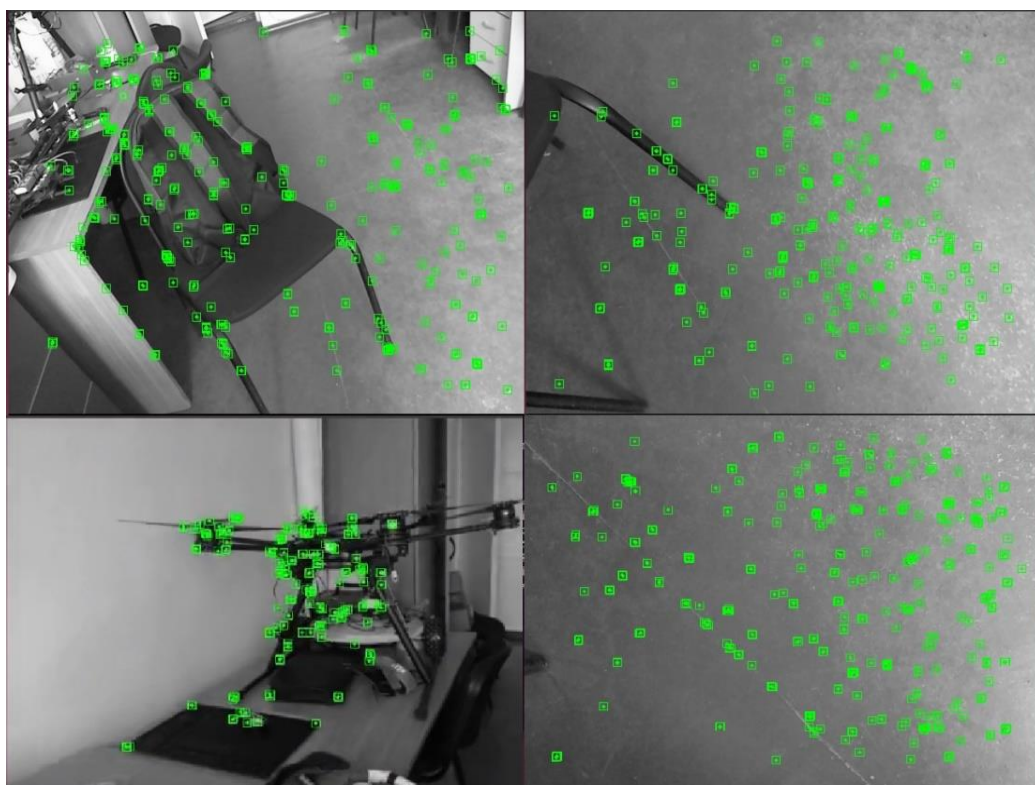


Рисунок 8.6 – Реализация работы ORB SLAM

Прямая разреженная одометрия (DSO). DSO является полностью прямым методом. Система использует оболочку для интеграции ROS и требует дополнительной разработки для реальной реализации робототехники. Основным преимуществом этой системы является плотная карта. DSO публикует положение робота как трансформацию. Система использовалась без замыкания петли, поэтому дрейф положения робота

накапливался во время первого круга и создавал двойные объекты среды на карте, когда робот делал второй круг. Также возможно, что трудности калибровки могут привести к дополнительным ошибкам в наших экспериментах. DSO создает плотную карту, и система является надежной с точки зрения отслеживания положений, но отсутствие замыкания петли добавляет на карту большой уровень шумов. Система, улучшенная обнаружением замыкания контура, даст более точные результаты при условии тщательной калибровки.

Полу-прямая визуальная одометрия (SVO). SVO имеет хорошую интеграцию с ROS. Получая информацию типа *msgs / Image*, SVO публикует геометрию *msgs / PoseWithCovarianceStamped* положения робота, облако точек, получаемое с датчика типа *msgs / PointCloud* для карты и *tf* для координатных преобразований. Система не является надежной для данного класса задач, поскольку она теряет отслеживание на поворотах робота, но может использоваться для получения дополнительной информации для прямых сегментов движения мобильного робота.

Плотное кусочно-параллельное отслеживание и картографирование (DPPTAM). DPPTAM - это прямой метод, использующийся для оценки положения мобильного робота. Данный метод реализован на базовом предположении, что области с одинаковым цветом принадлежат приблизительно к одной плоскости. Поскольку система потеряла отслеживание на поворотах робота, система недостаточно надежна для применения в помещении.

В результате работы был разработан прототип мобильного робота для проведения экспериментов. Робот был запущен в помещении по телеуправляемой замкнутой траектории вдоль известного периметра квадратной рабочей зоны, регистрируя данные телеметрических датчиков для автономной обработки. Таким образом, набор данных был собран от

бортовых датчиков визуальной информации и обработан с использованием следующих методов SLAM:

- 1) крупномасштабный прямой монокуляр SLAM (LSD SLAM);
- 2) ORB SLAM;
- 3) прямая разреженная одометрия (DSO);

Результаты выполнения для систем SLAM были исследованы и сравнены с использованием метрик.

Основные результаты исследований:

1) параллельное отслеживание и картографирование (PTAM), полупрямая визуальная одометрия (SVO), плотное кусочно-параллельное отслеживание и картирование (DPPTAM) провалили эксперименты, поскольку они потеряли текущее местоположение робота из-за отсутствия большого числа особых точек;

2) крупномасштабный прямой монокулярный SLAM (LSD SLAM), ORB SLAM, прямая разреженная одометрия (DSO) могут использоваться для решения проблемы локализации с дополнительным модулем восстановления масштаба;

3) ни одна монокулярная система SLAM не может справиться с проблемой неоднозначности шкалы масштаба без дополнительной информации о среде для её дальнейшего восстановления;

4) система Visual SLAM: карта RTAB продемонстрировала наилучшие результаты для проблемы локализации в наших экспериментах с RMSE ATE на 0.163 м, но она имеет проблему с прокладыванием пути вблизи монохромных стен.

Таблица 8.1 – Абсолютная ошибка траектории для различных систем

Метод	RMSE (m)	Max (m)	Min (m)
ORB-SLAM	0.166	0.257	0.047
LSD-SLAM	0.301	0.553	0.08
DSO	0.459	0.764	0.007

Наиболее надежной и стабильной из протестированных систем является ORB SLAM с RMSE ATE 0.190 м. Однако, производительность визуальных систем SLAM сильно зависит от вычислительных ресурсов мобильного робота.

8.5 Реализация и сравнение работы алгоритмов SLAM-навигации основанные на использовании лазерных дальномеров

На текущий момент в ROS реализованы два наиболее популярных программных метода для работы с данными лидара и дальнейшим построением карт на их основе.

Первый программный пакет получил название BLAM!. BLAM! написан на C ++ с некоторыми элементами интерфейса Python, скрытыми операционной системой ROS. Входные данные Lidar должны публиковаться в топик / velodyne_points, используя тип сообщения sensor_msgs :: PointCloud2. BLAM! Имеет два режима работы:

Для запуска в онлайн-режиме (например, путем воспроизведения bagfile с другого терминала или с использованием потока в реальном времени), используется команда:

```
roslaunch blam_example test_online.launch
```

Для запуска в автономном режиме, то есть путем загрузки файла bagfile и обработки его данных как можно быстрее, необходимо установить имя и

имя пакета в файле `blam_example / launch / test_offline.launch` и используется команда:

```
roslaunch blam_example test_offline.launch
```

Пример файла конфигурации `.rviz` предоставляется в файле `blam_example / rviz / lidar_slam.rviz`.

Данный программный пакет позволяет получать точные карты окружающей местности, а формат выходных карт хорошо совместим с программным пакетом Octomap, что позволяет без проблем обмениваться пакетами данных между двумя данными программами. Также достоинством данного программного пакета является встроенная функция «замыкания петель» (loop closure), что позволяет, при сопоставлении уже двух заведомо известных участков карты, отбрасывать огромное количество ложных измерений и производить уплотнение всего облака точек карты и перестраивать карту на основе данного уплотнения.

Однако главным минусом данного пакета является отсутствие канала получения для данных с IMU датчиков, другими словами данный программный пакет просто не использует данные с IMU. Из-за этого, несмотря на хорошее качество получаемых карт местности, BLAM! не дает возможности производить корректную оценку одометрии. Также, в связи с данным недостатком, нередко случаи разрушения целостности карт при потере данных в разреженных средах, либо в средах с низкой плотностью облака точек.

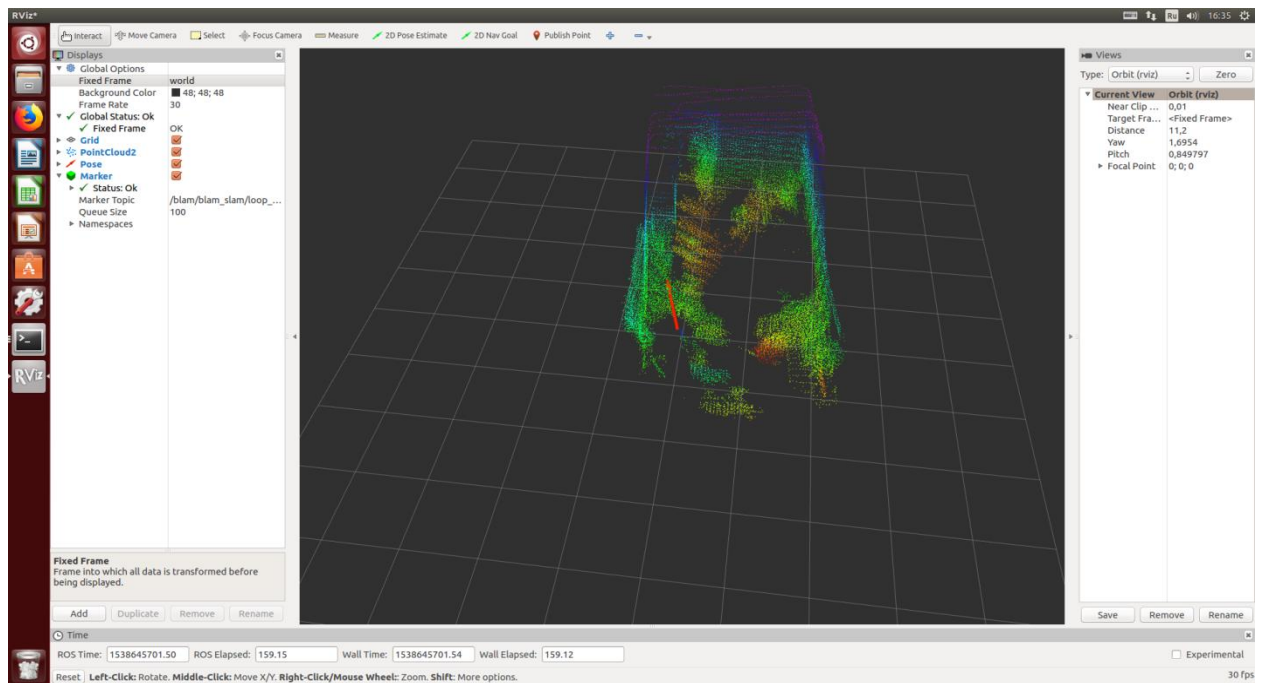


Рисунок 8.8 – Результат работы программного пакета BLAM! (базовый фрейм)

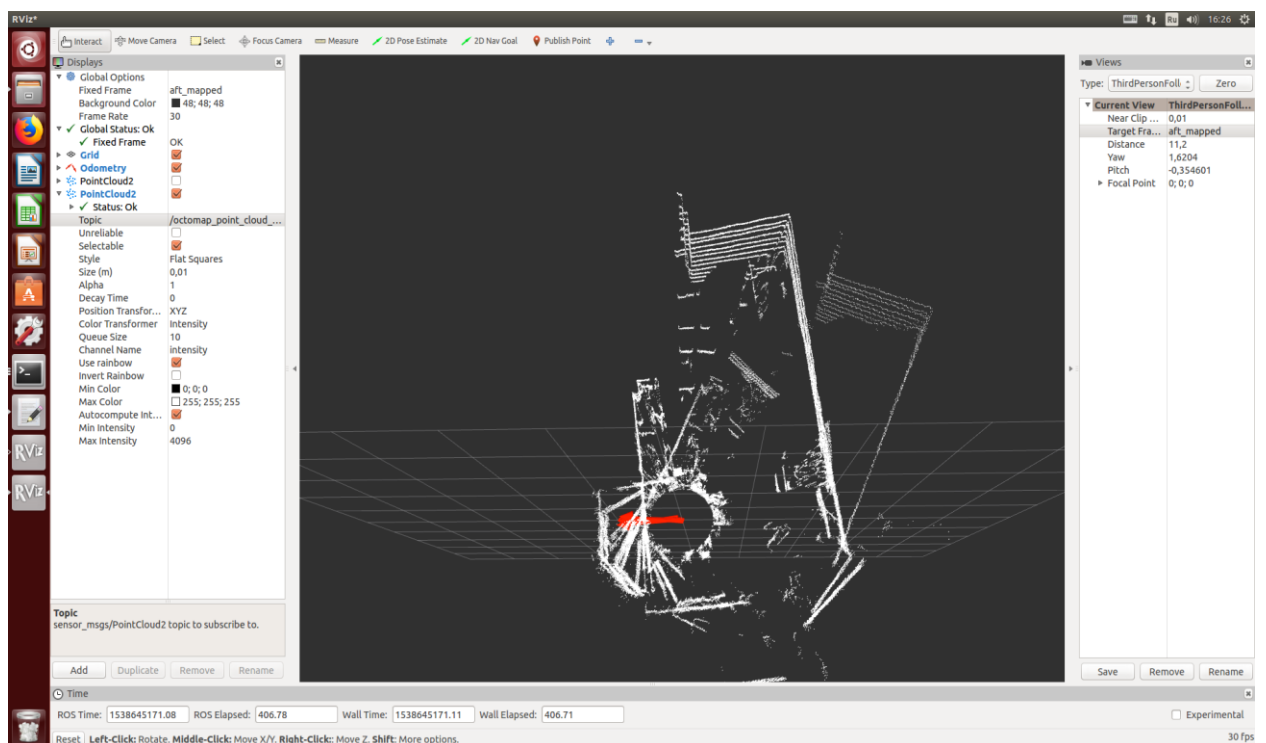


Рисунок 8.9 – Результат работы программного пакета BLAM! (смещение фрейма при отсутствии IMU)

Второй программный пакет именуется как LOAM (laser odometry and mapping). Laser Odometry and Mapping (Loam) - это метод реального времени

для оценки положения и картографирования с использованием 3D-лидара. Программа состоит из двух основных потоков, работающих параллельно. Поток одометрии вычисляет движение лидара между двумя развертками при более высокой частоте кадров. Он также устраняет искажения в облаке точек, вызванные движением лидара. Поток картографирования берет неискаженное облако точек и постепенно создает карту, и одновременно вычисляет позицию лидара на карте с более низкой частотой кадров. Оценка положения лидара представляет собой комбинацию выходов двух потоков.

Если доступен IMU, ориентация (интегрированная по угловой скорости) и измерения ускорения используются для решения общего движения лидара, в то время как программа использует данные о линейном движении.

Программа протестирована на ПК с четырьмя ядрами 2,5 ГГц и 12 ГБ памяти (программа потребляет два ядра). Также доступны еще две версии программы с использованием спинового лидара и непрерывного лидара.

На рисунках 8.8, 8.9 и 8.10 представлена работа алгоритма на основе данных с лидара VLP16 и данных с IMU, получаемых с полетного контроллера PixHawk2

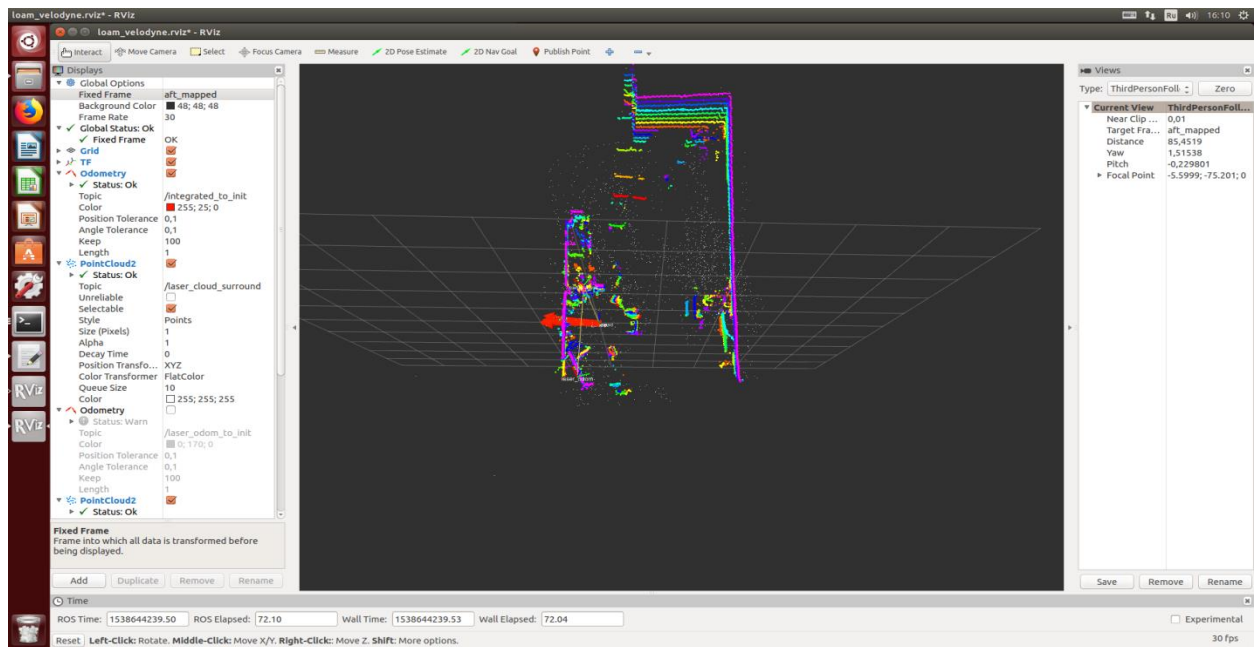


Рисунок 8.10 – Результат работы программного пакета LOAM (базовый фрейм)

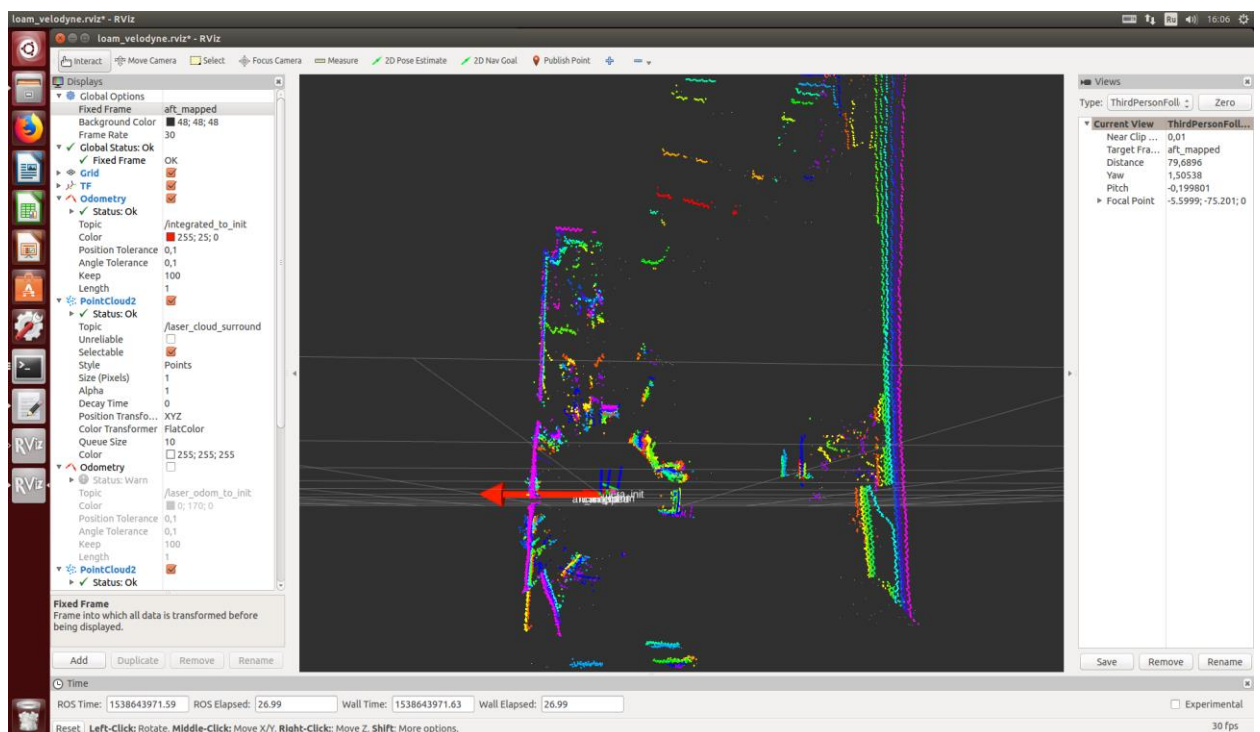


Рисунок 8.11 – Результат работы программного пакета LOAM

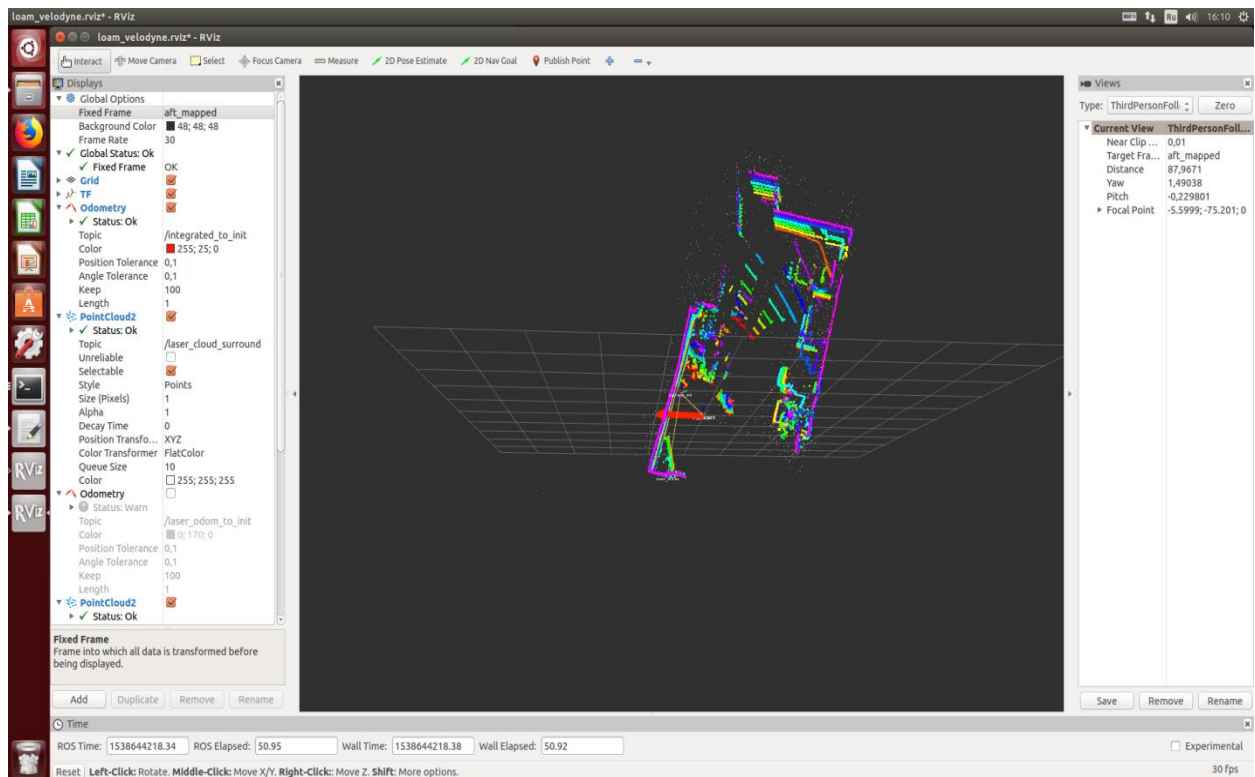


Рисунок 8.12 – Результат работы программного пакета LOAM (фрейм со смещением)

Как можно видеть на выше представленных рисунках при смещении лидара не происходит искажения карты, и конечный фрейм сохраняет целостность и, используя оба потока данных с VLP16 и IMU более точно обрабатывает вычисления конечного фрейма. Данный подход позволяет избежать серьезных ошибок при вычислении одометрии, однако в данном программном пакете отсутствует функция поиска замкнутых петель, из-за чего сохраняется некоторое число ложных измерений, а конечное облако точек карты получается менее плотным [58].

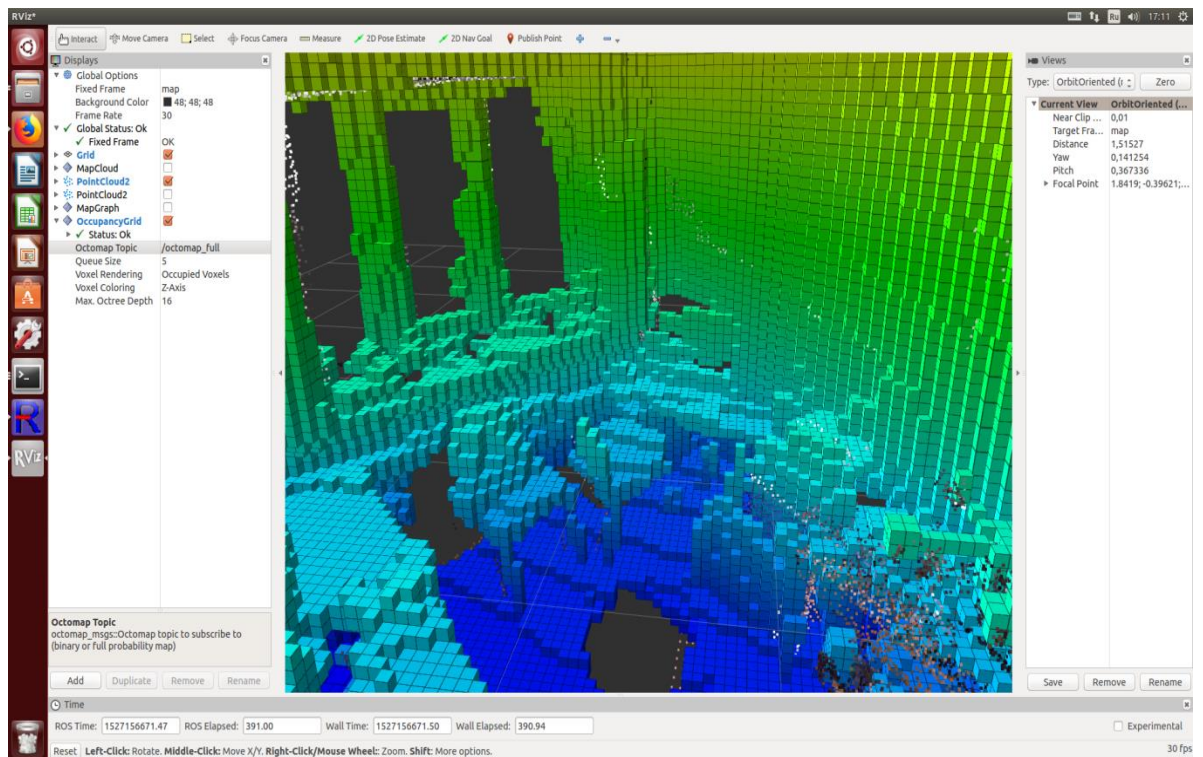


Рисунок 8.13 – Результат работы программного пакета LOAM (во время построения карты в режиме реального времени)

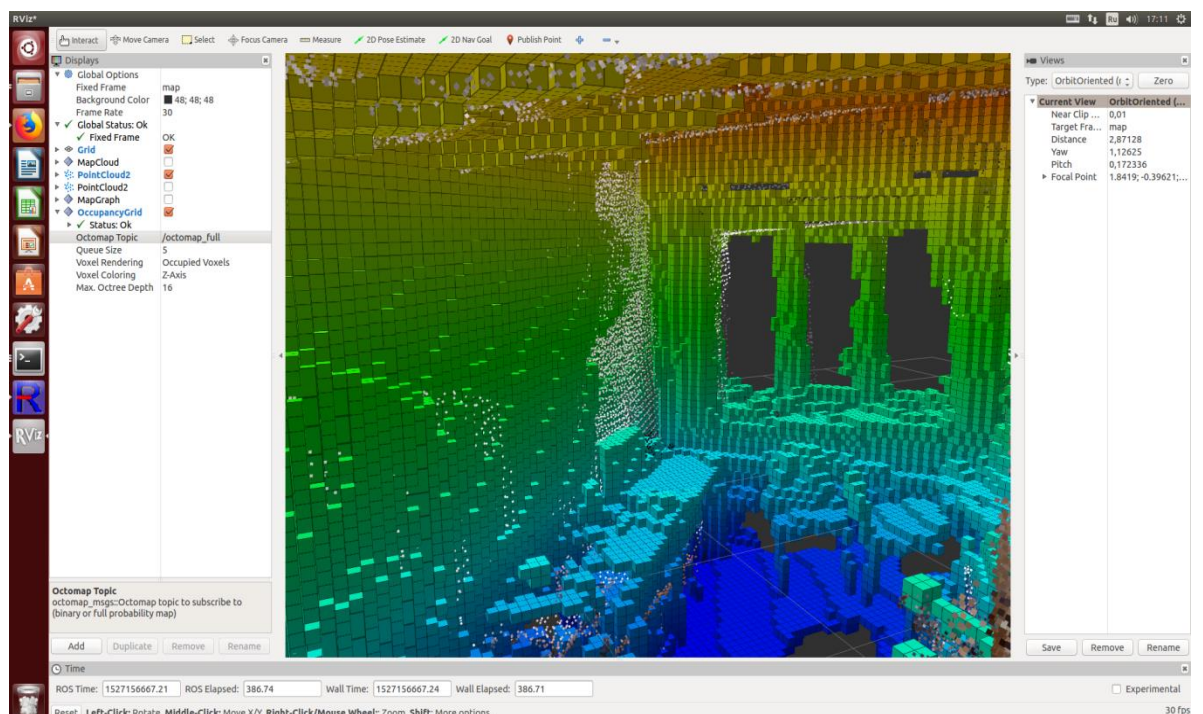


Рисунок 8.14 – Результат работы программного пакета LOAM (во время построения карты в режиме реального времени)

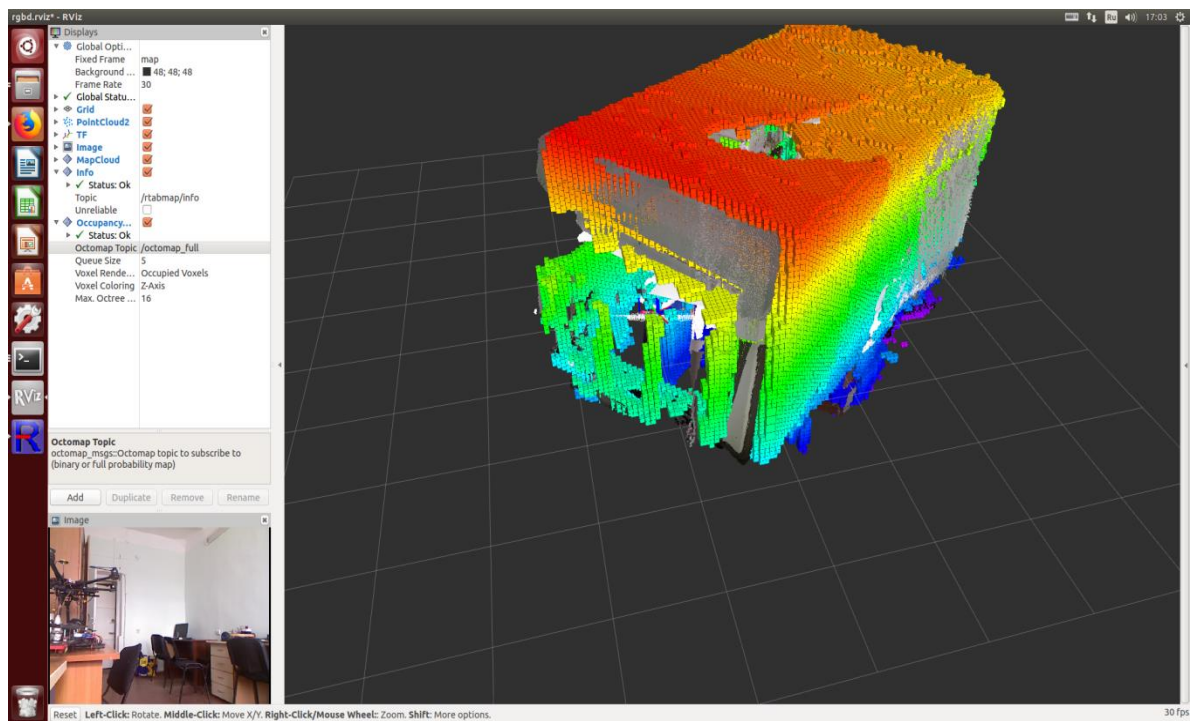


Рисунок 8.15 – Работа с полученным облаком точек для дальнейшей обработки в пакете Octomap

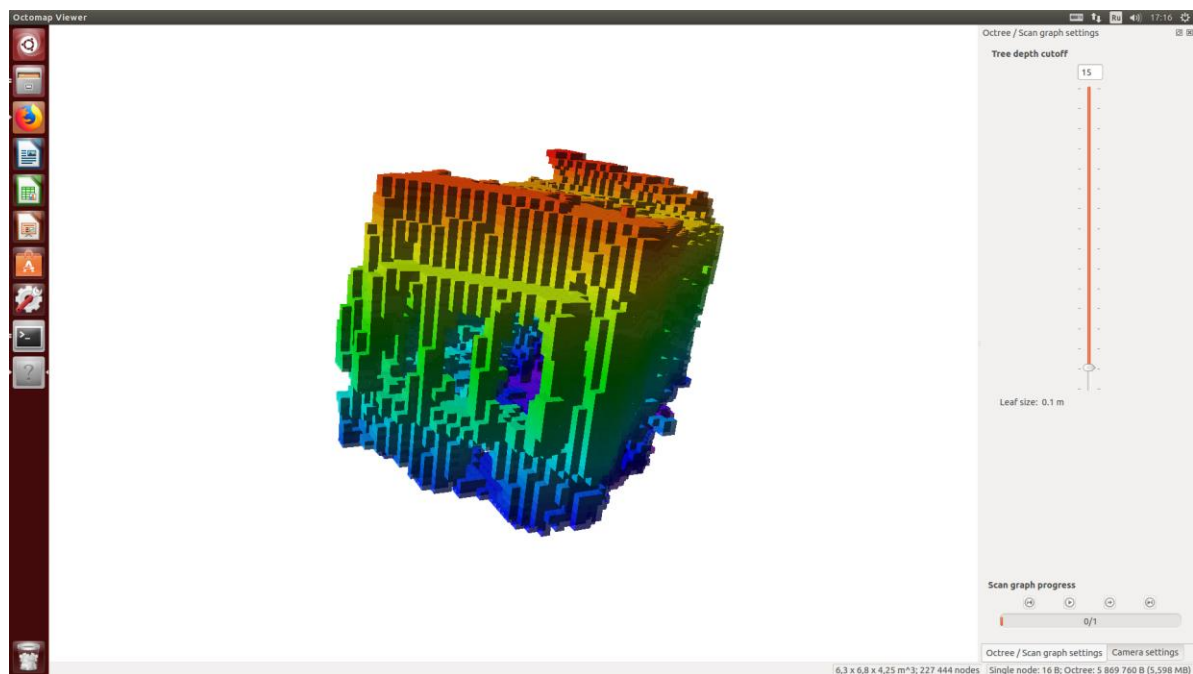


Рисунок 8.16 – Работа с облаком точек в программном пакете Octomap (базовый фрейм пакета Octovis)

8.6 Интеграция фреймворка Google Cartographer с алгоритмами Obstacle Avoidance

Различные навигационные решения, основанные на интеграции INS / GPS для мобильных роботов, были разработаны и протестированы во многих коммерческих системах автопилотирования БПЛА. Тем не менее, существующие навигационные решения не могут обеспечить навигацию в условиях, где отсутствует GPS или в условиях плохой связи со спутниками, только потому, что GPS необходим для компенсации дрейфа из-за смещений и шума в измерениях IMU (акселерометр и гироскоп). В случае систем вертолетов отсутствие GPS в течение даже коротких промежутков времени может привести к смещению ориентации, что затрудняет использование автономных систем управления транспортным средством данного типа. Аналогичным образом, дрейф может возникнуть из-за магнитных помех при пролете вблизи электрических проводов. Навигационные системы, устойчивые к таким сбоям, в крайней степени необходимы для обеспечения безопасного и надежного автономного полета, особенно если мобильные роботы используются в районах, расположенных вблизи зданий и людей.

Одной из критических проблем в автономной навигации мобильных роботов в неопределенных условиях (например, в городской местности) является обход препятствий, при котором робототехническая система должна обнаруживать, отслеживать и избегать препятствия на траектории полета. Активные датчики, такие как лидар, наиболее подходят для этой цели, однако они имеют значительный вес, стоимость и энергопотребление.

Поскольку большинство небольших БПЛА оснащены монокулярной камерой и инерциальным измерительным блоком (IMU), использование этих датчиков для предотвращения столкновений с препятствиями может привести к минимальной нагрузке по весу. Таким образом, полет в сценарии, где GPS может периодически теряться, требует подхода, который может

беспрепятственно включать информацию от IMU, СТЗ и других вспомогательных датчиков.

Во многих алгоритмах по поиску и предотвращению столкновения с препятствиями информация от датчиков визуальной информации, такая как расхождение поля данных, фокус расширения и время контакта, использовалась в качестве основного базиса для отклонения от траектории. Однако подходы, основанные на системах технического зрения визуальной информации, обычно просто пытаются избежать областей с высоким оптическим потоком. Также, они не обеспечивают точного вычисления местоположения транспортного средства или картографирования окружающей среды, что важно для автономной навигации в долгосрочной перспективе.

SLAM решает проблему автономной локализации транспортного средства с минимальным использованием / без априорной информации об окружающей среде. SLAM алгоритмы были успешно протестированы и реализованы на многих мобильных робототехнических системах. Перевод таких решений на бортовые мобильные системы, тем не менее, остается сложной задачей.

Современные реализации подхода VSLAM решают проблему инициализации функции гораздо более быстрым способом (не полагаясь на стерео движения или другие методы отложенной инициализации), опираясь на так называемую обратную параметризацию глубины характерных точек. Использование подхода ICP, как правило, приводит к тому, что реализация фильтра становится проще, и отслеживание результатов обработки функций может начинаться с первой итерации фильтра. Одним из важных недостатков подхода VSLAM является то, что первоначальное предположение о глубине объекта может вносить смещение в масштаб карты, которую алгоритм строит с течением времени. Благодаря интеграции данных, полученных с

инерциального датчика и метода ICP, влияние начального измерения глубины в масштабе карты может быть уменьшено.

Была проведена попытка использовать MonoSLAM для обхода препятствий в замкнутом пространстве с помощью оценок обратной связи из MonoSLAM в алгоритме управления транспортным средством. Одним критическим недостатком традиционного (основанного на точечных признаках) подхода MonoSLAM с точки зрения экономии вычислительной мощности, используемой для расчётов, в дальнейшем используемых для предотвращения столкновений с препятствиями, является то, что решение SLAM производит построение достаточно разреженной карты окружающей среды. Плотность карты может быть увеличена путем добавления большего количества характерных точек в фильтр SLAM. Однако увеличение количества характерных точек также значительно увеличивает вычислительную нагрузку алгоритма SLAM. Чтобы решить эту проблему, было решено использовать LidarSLAM и сенсор Velodyne VLP16 при совместном использовании ограниченного числа характерных точек в векторе состояний SLAM.

Было решено объединить 3D LidarSLAM с инерциальной системой и сегментацией изображения, чтобы идентифицировать препятствия и построить плотную карту вокруг него. Основное преимущество этого подхода заключается в том, что обработка всех данных интегрирована в SLAM. Сначала мы сегментируем данный входной поток как способ абстрагировать всю сцену. Затем контур сегмента извлекается и определяются угловые точки сегмента в схеме. Мы сохраняем основанную на точках структуру LidarSLAM, используя ограниченное количество точек для получения быстрой и надежной оценки. Чтобы улучшить картографическую составляющую SLAM, был протестирован отдельный процесс, основанный на сегментах, а не на точках. Подчеркнутый угол отдельного сегмента

данных, соответствующего препятствию в трехмерном мире, извлекается для дальнейшего использования.

Поскольку сегментация данных увеличивает вычислительную нагрузку на систему, было принято решение использовать быструю сегментацию, основанную на упрощенной математической модели.

Google Cartographer - это фреймворк, который обеспечивает одновременную локализацию и картографирование в реальном времени (SLAM) в 2D и 3D форматах на различных платформах и различных конфигурациях датчиков.

Google Cartographer с ROS используется в качестве программного стека с некоторыми изменениями в коде в зависимости от выбора оборудования.

Аппаратное обеспечение:

1) Velodyne VLP-16. Он используется для измерения расстояния до цели путем облучения цели импульсным лазерным светом и измерения отраженных импульсов датчиком. Различия во времени возврата и длине волны лазера можно, затем, использовать для создания цифрового трехмерного представления цели;

2) Pixhawk PX4 (Cube) - это контроллер автопилотирования с открытым исходным кодом, ориентированный на недорогие автономные БПЛА. Именно полётный контроллер является центральным устройством управления движением мобильного робота;

3) NVIDIA Jetson AGX Xavier - это полнофункциональная одноплатная платформа для произведения всех вычислений. Она поставляется со средой Linux, которая включает поддержку многих распространенных API-интерфейсов и поддерживается всей цепочкой инструментов разработки NVIDIA. Здесь Jetson используется в качестве основной платы контроллера, который хранит и выполняет весь программный стек (ROS и Google

Cartographer), а также реализует обмен данными между Pixhawk и датчиком Velodyne;

4) Veyron 2x25a - это контроллер двигателей, который реализует запуск двигателей на основании ШИМ-сигналов поступающих с полётного контроллера;

5) моторы - 2 мотора используются в конфигурации R2D2 с гусеничными лентами для эффективного и разнонаправленного движения робота.

Первым шагом является загрузка и компиляция всех библиотек ROS в Jetson с официального сайта ROS. После компиляции Google Cartographer необходимо загрузить и скомпилировать, разрешив при этом устранять разорванные зависимости между программными пакетами.

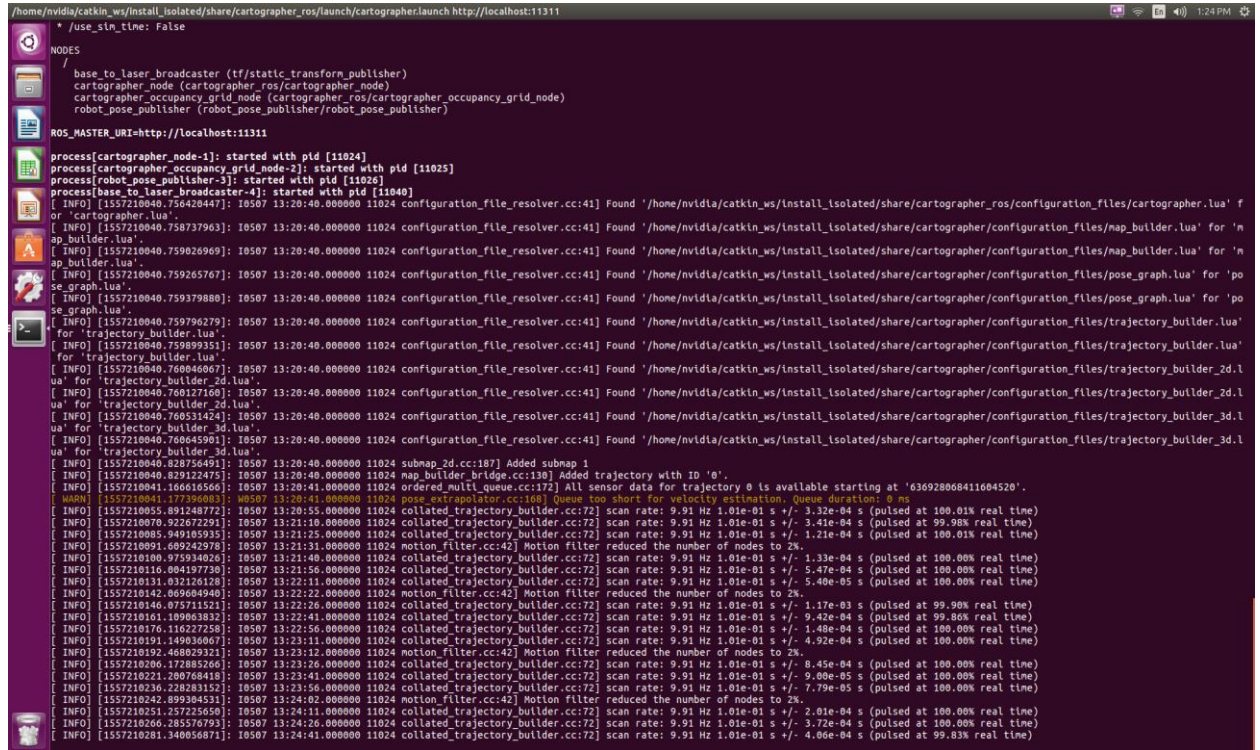
Следующим шагом является калибровка полетного контроллера (pixhawk) и проверка данных карты (тема: `/velodyne_points`) из датчика Velodyne. Как только все проверено и откалибровано, контроллер полета должен быть переведен в управления режим Guided, после которого он должен быть переведён в активный режим управления.

Управляемый режим помогает нам отправлять команды в виде целей на контроллер полета, и контроллер автоматически перемещается по простым целям. Начальная настройка завершена, и теперь робот готов к тестированию.

Для тестирования используется другой компьютер для удаленного доступа к графическому интерфейсу Jetson (необязательно). Для визуализации данных карты должен быть запущен графический интерфейс Rviz, должны быть инициализированы все данные карты из топиков (`/map`, `/point_cloud2`, `/pose`), чтобы сформировать визуальную карту и оценку положение в 3D пространстве.

Последний шаг - дать роботу цель навигации, используя кнопку «2D Nav Goal» в пакете Rviz с дальнейшей проверкой точности позиционирования робота визуальным методом.

Результаты работы представлены на рисунках 8.17 – 8.21.



```

/home/nvidia/catkin_ws/install_isolated/share/cartographer_ros/launch/cartographer.launch http://localhost:11311
* /use_sim_time: False

NODES
  /
    base_to_laser_broadcaster (tf/static_transform_publisher)
    cartographer_node (cartographer_ros/cartographer_node)
    cartographer_occupancy_grid_node (cartographer_ros/cartographer_occupancy_grid_node)
    robot_pose_publisher (robot_pose_publisher/robot_pose_publisher)

ROS_MASTER_URI=http://localhost:11311

process[cartographer_node-1]: started with pid [11024]
process[cartographer_occupancy_grid_node-2]: started with pid [11025]
process[robot_pose_publisher-3]: started with pid [11026]
process[base_to_laser_broadcaster-4]: started with pid [11040]
[ INFO] [1557210040.756420447]: 10507 13:20:40.000000 11024 configuration_file_resolver.cc:41] Found '/home/nvidia/catkin_ws/install_isolated/share/cartographer_ros/configuration_files/cartographer.lua' for 'cartographer.lua'.
[ INFO] [1557210040.758737963]: 10507 13:20:40.000000 11024 configuration_file_resolver.cc:41] Found '/home/nvidia/catkin_ws/install_isolated/share/cartographer/configuration_files/map_builder.lua' for 'map_builder.lua'.
[ INFO] [1557210040.759026969]: 10507 13:20:40.000000 11024 configuration_file_resolver.cc:41] Found '/home/nvidia/catkin_ws/install_isolated/share/cartographer/configuration_files/map_builder.lua' for 'map_builder.lua'.
[ INFO] [1557210040.759265767]: 10507 13:20:40.000000 11024 configuration_file_resolver.cc:41] Found '/home/nvidia/catkin_ws/install_isolated/share/cartographer/configuration_files/pose_graph.lua' for 'pose_graph.lua'.
[ INFO] [1557210040.759379880]: 10507 13:20:40.000000 11024 configuration_file_resolver.cc:41] Found '/home/nvidia/catkin_ws/install_isolated/share/cartographer/configuration_files/pose_graph.lua' for 'pose_graph.lua'.
[ INFO] [1557210040.759796279]: 10507 13:20:40.000000 11024 configuration_file_resolver.cc:41] Found '/home/nvidia/catkin_ws/install_isolated/share/cartographer/configuration_files/trajectory_builder.lua' for 'trajectory_builder.lua'.
[ INFO] [1557210040.760127160]: 10507 13:20:40.000000 11024 configuration_file_resolver.cc:41] Found '/home/nvidia/catkin_ws/install_isolated/share/cartographer/configuration_files/trajectory_builder.lua' for 'trajectory_builder.lua'.
[ INFO] [1557210040.76046067]: 10507 13:20:40.000000 11024 configuration_file_resolver.cc:41] Found '/home/nvidia/catkin_ws/install_isolated/share/cartographer/configuration_files/trajectory_builder_2d.lua' for 'trajectory_builder_2d.lua'.
[ INFO] [1557210040.760812716]: 10507 13:20:40.000000 11024 configuration_file_resolver.cc:41] Found '/home/nvidia/catkin_ws/install_isolated/share/cartographer/configuration_files/trajectory_builder_2d.lua' for 'trajectory_builder_2d.lua'.
[ INFO] [1557210040.760531424]: 10507 13:20:40.000000 11024 configuration_file_resolver.cc:41] Found '/home/nvidia/catkin_ws/install_isolated/share/cartographer/configuration_files/trajectory_builder_3d.lua' for 'trajectory_builder_3d.lua'.
[ INFO] [1557210040.760645901]: 10507 13:20:40.000000 11024 configuration_file_resolver.cc:41] Found '/home/nvidia/catkin_ws/install_isolated/share/cartographer/configuration_files/trajectory_builder_3d.lua' for 'trajectory_builder_3d.lua'.
[ INFO] [1557210040.828756491]: 10507 13:20:40.000000 11024 submap_2d.cc:107] Added submap 1
[ INFO] [1557210040.82912475]: 10507 13:20:40.000000 11024 map_builder_bridge.cc:138] Added trajectory with ID '0'.
[ INFO] [1557210041.166610566]: 10507 13:20:41.000000 11024 ordered_multi_queue.cc:172] All sensor data for trajectory 0 is available starting at '636928068411604520'.
[ WARN] [1557210041.177396083]: 10507 13:20:41.000000 11024 pose_extrapolator.cc:168] Queue too short for velocity estimation. Queue duration: 0 ms
[ INFO] [1557210055.891248772]: 10507 13:20:55.000000 11024 collated_trajectory_builder.cc:72] scan rate: 9.91 Hz 1.01e-01 s +/- 3.32e-04 s (pulsed at 100.01% real time)
[ INFO] [1557210070.922672291]: 10507 13:21:10.000000 11024 collated_trajectory_builder.cc:72] scan rate: 9.91 Hz 1.01e-01 s +/- 3.41e-04 s (pulsed at 99.98% real time)
[ INFO] [1557210085.949105935]: 10507 13:21:25.000000 11024 collated_trajectory_builder.cc:72] scan rate: 9.91 Hz 1.01e-01 s +/- 1.21e-04 s (pulsed at 100.01% real time)
[ INFO] [1557210091.609242978]: 10507 13:21:31.000000 11024 motion_filter.cc:42] Motion filter reduced the number of nodes to 2k.
[ INFO] [1557210100.975934026]: 10507 13:21:40.000000 11024 collated_trajectory_builder.cc:72] scan rate: 9.91 Hz 1.01e-01 s +/- 1.33e-04 s (pulsed at 100.00% real time)
[ INFO] [1557210116.064197730]: 10507 13:21:56.000000 11024 collated_trajectory_builder.cc:72] scan rate: 9.91 Hz 1.01e-01 s +/- 5.47e-04 s (pulsed at 100.00% real time)
[ INFO] [1557210131.032126128]: 10507 13:22:11.000000 11024 collated_trajectory_builder.cc:72] scan rate: 9.91 Hz 1.01e-01 s +/- 5.40e-05 s (pulsed at 100.00% real time)
[ INFO] [1557210142.069064940]: 10507 13:22:22.000000 11024 motion_filter.cc:42] Motion filter reduced the number of nodes to 2k.
[ INFO] [1557210146.075711521]: 10507 13:22:26.000000 11024 collated_trajectory_builder.cc:72] scan rate: 9.91 Hz 1.01e-01 s +/- 1.17e-03 s (pulsed at 99.90% real time)
[ INFO] [1557210161.109063832]: 10507 13:22:41.000000 11024 collated_trajectory_builder.cc:72] scan rate: 9.91 Hz 1.01e-01 s +/- 9.42e-04 s (pulsed at 99.86% real time)
[ INFO] [1557210176.116227558]: 10507 13:22:56.000000 11024 collated_trajectory_builder.cc:72] scan rate: 9.91 Hz 1.01e-01 s +/- 1.48e-04 s (pulsed at 100.00% real time)
[ INFO] [1557210191.149036007]: 10507 13:23:11.000000 11024 collated_trajectory_builder.cc:72] scan rate: 9.91 Hz 1.01e-01 s +/- 4.92e-04 s (pulsed at 100.00% real time)
[ INFO] [1557210192.468029321]: 10507 13:23:12.000000 11024 motion_filter.cc:42] Motion filter reduced the number of nodes to 2k.
[ INFO] [1557210206.172885266]: 10507 13:23:26.000000 11024 collated_trajectory_builder.cc:72] scan rate: 9.91 Hz 1.01e-01 s +/- 8.45e-04 s (pulsed at 100.00% real time)
[ INFO] [1557210221.280768418]: 10507 13:23:41.000000 11024 collated_trajectory_builder.cc:72] scan rate: 9.91 Hz 1.01e-01 s +/- 9.00e-05 s (pulsed at 100.00% real time)
[ INFO] [1557210236.228283152]: 10507 13:23:56.000000 11024 collated_trajectory_builder.cc:72] scan rate: 9.91 Hz 1.01e-01 s +/- 7.79e-05 s (pulsed at 100.00% real time)
[ INFO] [1557210242.899304531]: 10507 13:24:02.000000 11024 motion_filter.cc:42] Motion filter reduced the number of nodes to 2k.
[ INFO] [1557210251.257225650]: 10507 13:24:11.000000 11024 collated_trajectory_builder.cc:72] scan rate: 9.91 Hz 1.01e-01 s +/- 2.01e-04 s (pulsed at 100.00% real time)
[ INFO] [1557210266.285576793]: 10507 13:24:26.000000 11024 collated_trajectory_builder.cc:72] scan rate: 9.91 Hz 1.01e-01 s +/- 3.72e-04 s (pulsed at 100.00% real time)
[ INFO] [1557210281.348050871]: 10507 13:24:41.000000 11024 collated_trajectory_builder.cc:72] scan rate: 9.91 Hz 1.01e-01 s +/- 4.06e-04 s (pulsed at 99.83% real time)

```

Рисунок 8.17 – Результат работы фреймворка Google Cartographer

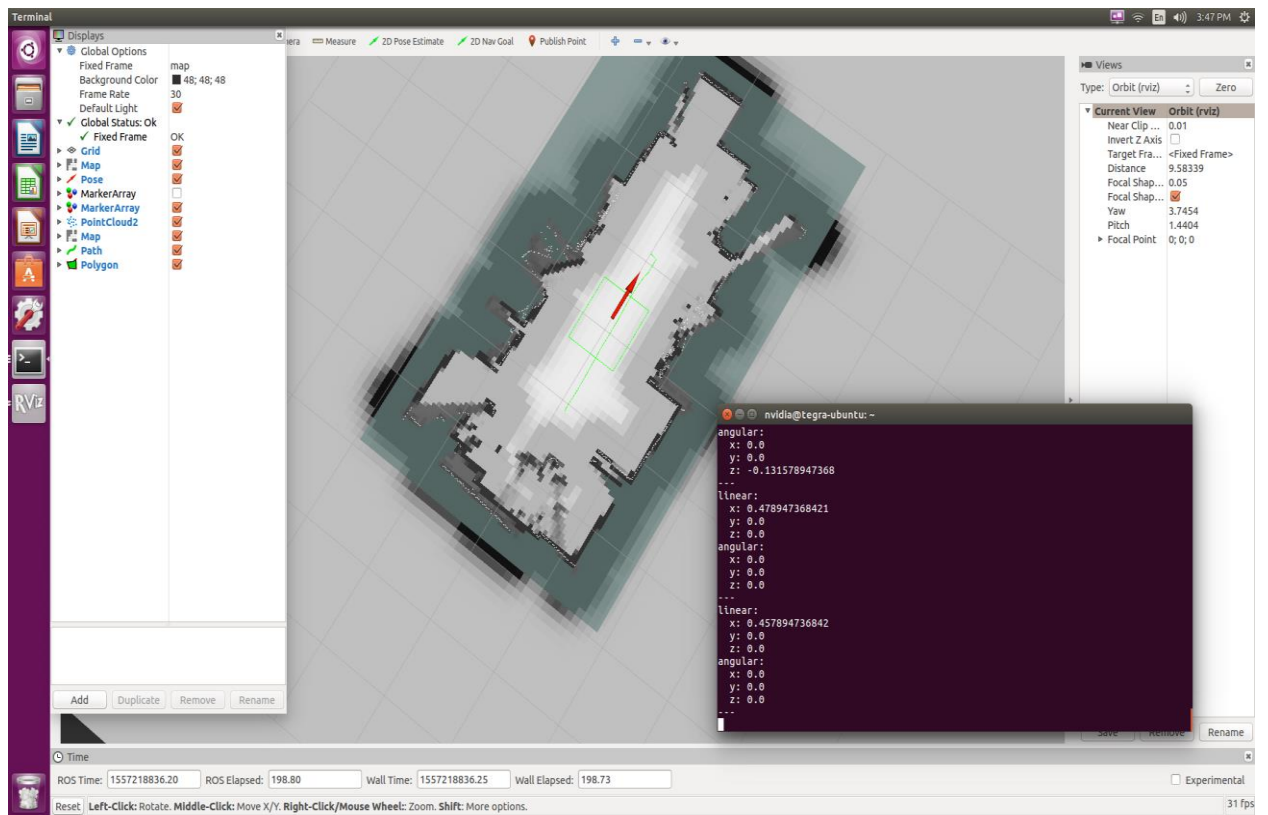


Рисунок 8.18 – Результат работы тестируемого фреймворка Google Cartographer. Визуальное отображение полученной 2D карты. Навигационная цель задана.



Рисунок 8.19 – Работа тестируемого модуля телеуправления

Результаты работы интегрированного алгоритма обхода препятствий представлены на рисунках 8.22 – 8.23. При изначально неподвижной мобильной платформе была задана навигационная цель. Затем для тестирования работы в режиме реального времени были выбраны два участка местности, в которых будет устанавливаться препятствие. Препятствие устанавливалось последовательно в каждой из двух точек. Алгоритм выполняет перестройку пути в режиме реального времени с учетом конечной навигационной цели, собственных геометрических размеров, а также на основании структуры управления мобильным роботом.

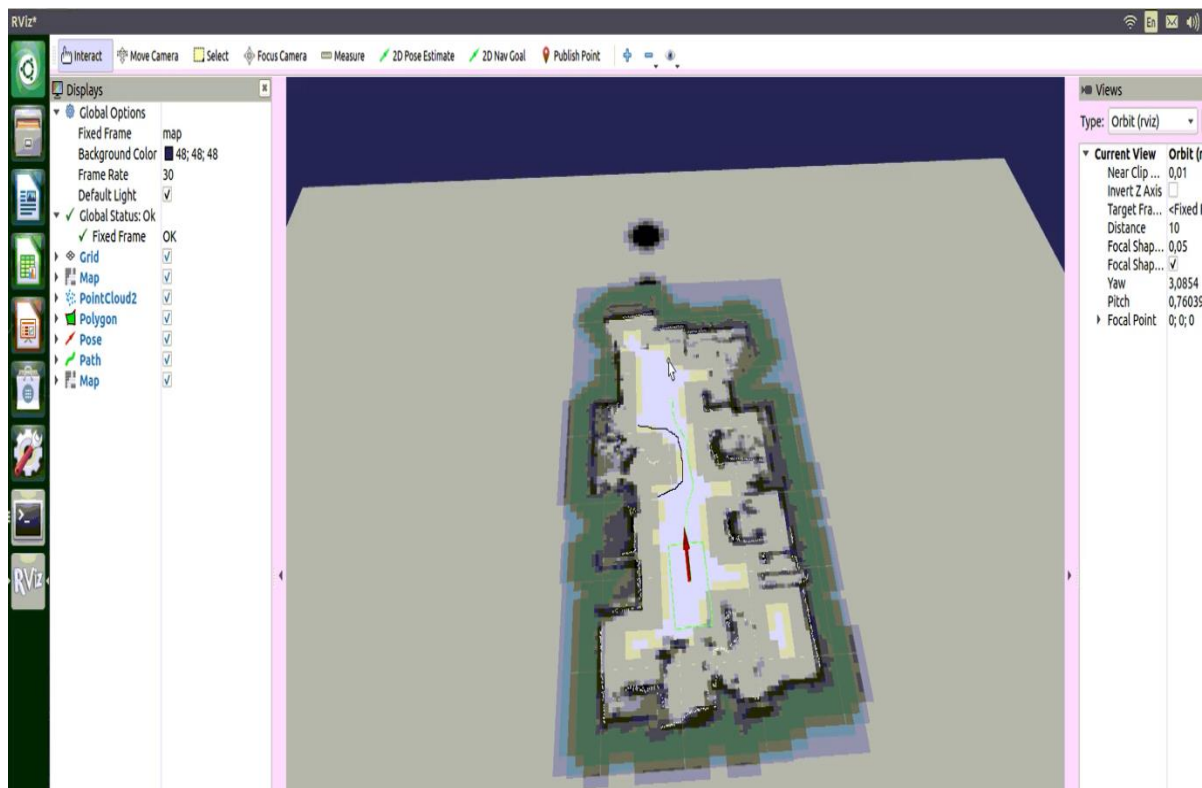


Рисунок 8.22 – Результат алгоритма Obstacle Avoidance. Препятствие установлено слева

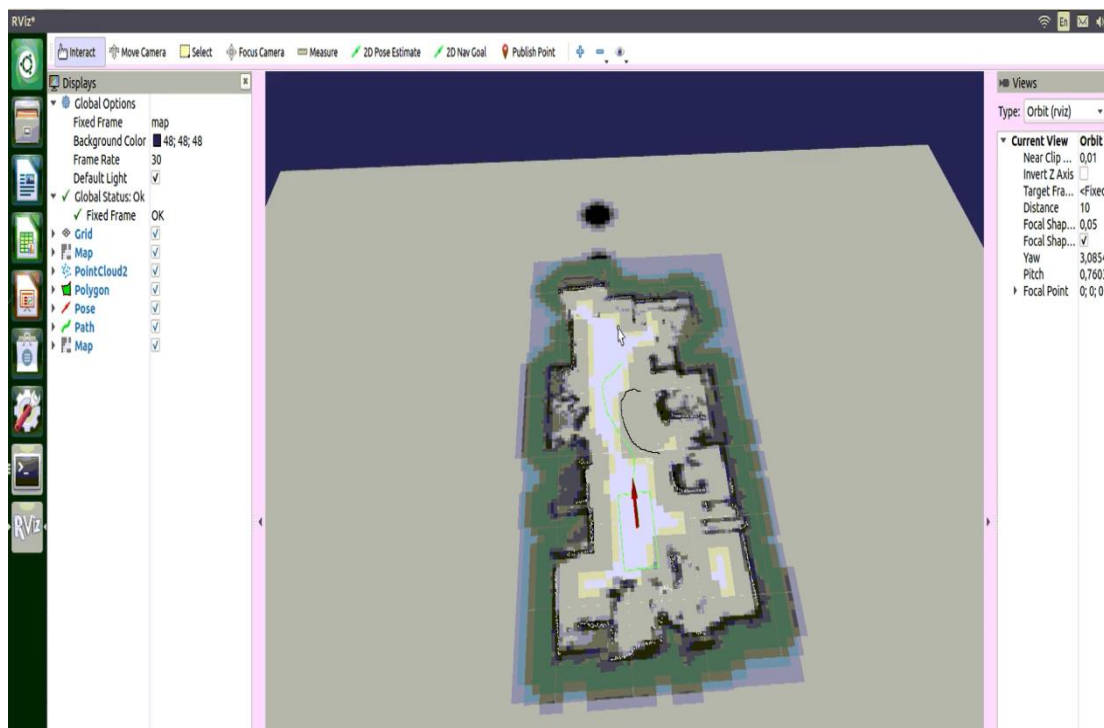


Рисунок 8.23 – Результат алгоритма Obstacle Avoidance. Препятствие установлено справа

Препятствия на полученной карте выделяются в серых тонах. За счет применения сегментации, в целях упрощения сложности вычислений, мелкие препятствия объединяются в единый псевдо объект, а все препятствия, при этом, выделяются фигурами простых геометрических форм, такими как линии, квадраты и прямоугольники. При пристраивании пути, алгоритм учитывает текущее положение мобильного робота, собственные геометрические размеры, геометрические размеры препятствий, а также относительный масштаб карты окружающей местности.

Итоговая функциональная схема работы данных пакетов в ROS, представлена на рисунке 8.24.

9 Реализация синтезированной модели

В результате запуска подсистемы управления оператор получает о списке и функциональном назначении узла *mavros_node*, узел *mavros* выводит полную информацию о состоянии подключения полётного контроллера в операционной системе и списке исполняемых им команд в зависимости от управления оператора в режиме реального времени. Также на экран оператора выводится информация о соответствующих каждой исполняемой команде микроконтроллера значениях векторов линейной и угловой скорости.

Данные значений векторов для вывода на экран оператора формируются в следующем формате:

linear:

x: value

y: value

z: value

anguar:

x: value

y: value

z: value

где *x*, *y*, *z* – обозначения соответствующих координат, *value* – текущее значение компоненты вектора.

Следующим этапом после запуска подсистемы телеуправления является реализация модели построения 2D и 3D карт местности.

Во время движения прототипа мобильного робота, на пути его следования могут возникать различные объекты-препятствия. Система технического зрения должна «обнаружить» препятствие, определить расстояние от платформы до объекта, выступающего препятствием. Логика используемого фреймворка Google Cartographer совместно с алгоритмом

LOAM для обнаружения неизвестного объекта-препятствия и вычисления расстояния до него, строится на анализе облака точек глубины. Так как основной датчик – Velodyne VLP16, на систему накладываются ограничения связанные с диапазоном «рабочих» расстояний самого сенсора.

В версии VLP16 для операционной системы Linux существует два режима работы сенсора:

- диапазон по умолчанию;
- секторный диапазон (Phase Lock).

При использовании режима работы сенсора Default Range, система будет способна обнаружить объект-препятствие, который находится от сенсора на расстояние не ближе чем на 0.05 метра и не дальше чем на 100 метров.

На рисунке 9.1 представлена блок-схема алгоритма формирования облака точек глубины подсистемы навигации.

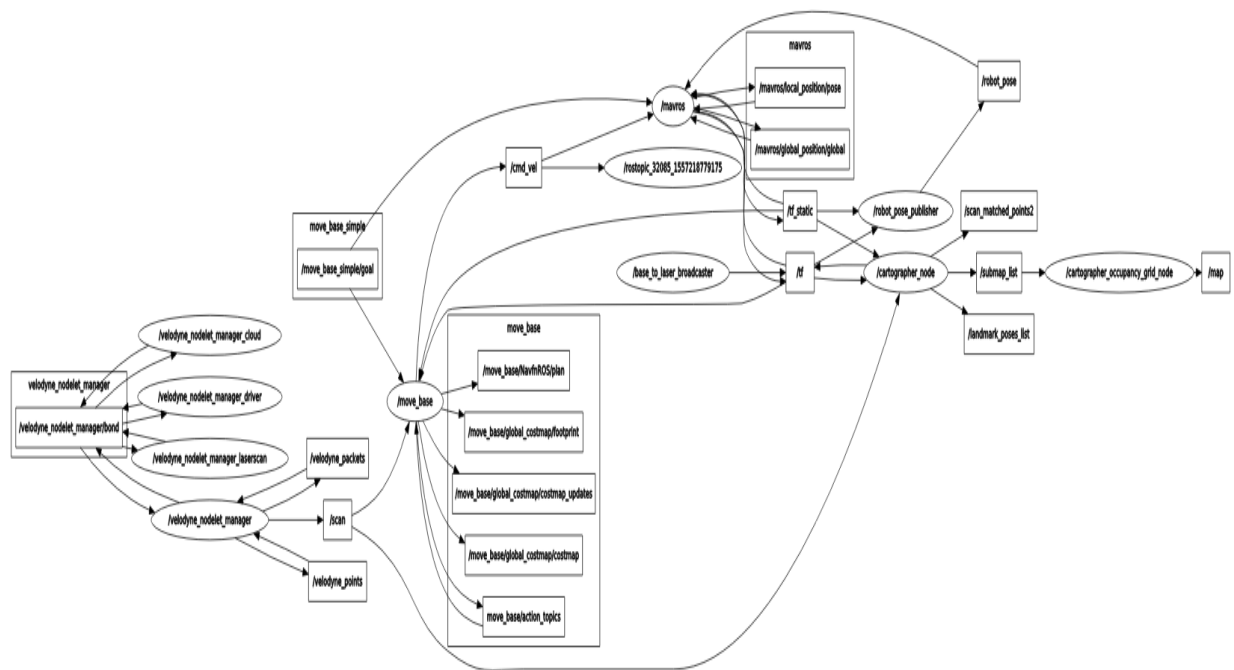


Рисунок 9.1 – Блок-схема алгоритма формирования облака точек глубины

В качестве основных элементов блок-схемы в пакете ROS представлены узлы исполняемых пакетов программ. На блок-схеме узлы и

формируемые ими темы объединены в прямоугольные рамки. Все темы внутри определенного узла выделены овальными рамками. Стрелками показано направление между темами соответствующих узлов.

Модель построения 3D и 2D карт определяется набором команд запуска и настройки пакетов ROS в операционной системе.

Настройка пакетов производится путем редактирования исполняемых файлов-скриптов. Данные файлы-скрипты реализованы на языке XML и могут быть отредактированы в любом текстовом редакторе. Для реализации построения 3D и 2D карт в программном пакете RViz требуется найти в среде ROS файл `cartographer.launch`, соответствующий запуску выбранного нами алгоритма решения SLAM.

Далее, для запуска построения 3D карты требуется выполнить следующий набор команд, каждую команду требуется запускать в отдельном терминале операционной системы Linux:

```
roslaunch velodyne_pointcloud VLP16.launch
```

```
roslaunch cartographer_ros cartographer.launch
```

Данный набор команд, согласно настройкам файла запуска, инициализирует запуск программных пакетов, контролирующих работу датчика, а также программных пакетов Google Cartographer и RViz.

На рисунке 9.2 показан результат работы алгоритма и построения 3D карты в пакете Rviz.

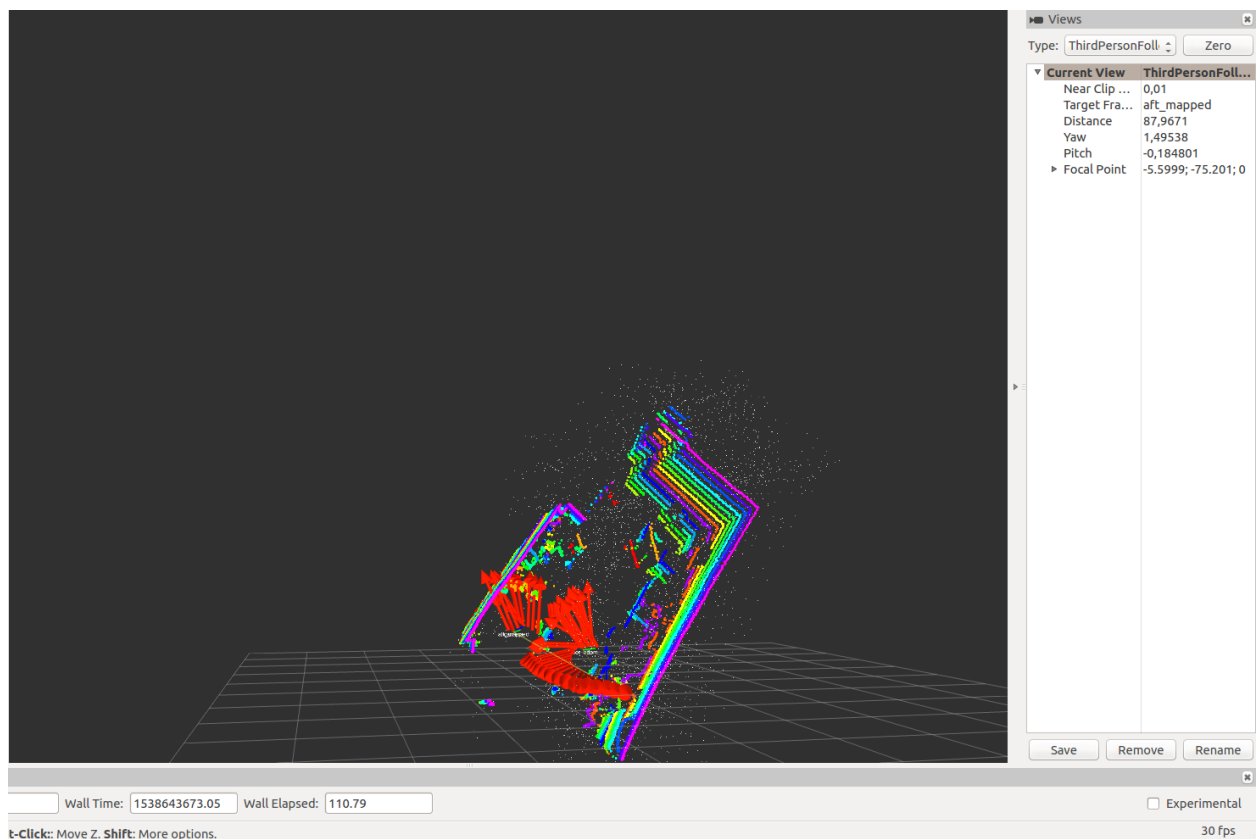


Рисунок 9.2 – Результат работы алгоритма в пакете RViz

На данном изображении видно, что пакет, на основе заданного алгоритма, вычисляет и формирует облако точек глубины и по полученным значениям формирует 3D карту местности. Белыми точками на линии отмечаются места, в которых, в область сенсора попадают новые, ранее не сканированные, области местности и происходит вычисление и дополнение облака точек глубины. Область, выделенная цветной градацией от синего до красного, является областью пространства сканируемой датчиком в данный момент.

Следующим этапом является формирование 2D карты местности. Формирование двухмерной карты происходит на базе полученного 3D облака точек глубины. Для формирования 2D карты в пакете RViz требуется добавить блок Мар, в настройках которого в графе Topic, который отвечает за выбор источника входных данных, указать источник потока данных

/cartographer/occupancy_grid_node, формирующийся в результате работы пакета Google Cartographer.

На рисунках 9.3 и 9.4 показан результат формирования 2D карты в пакете RViz.

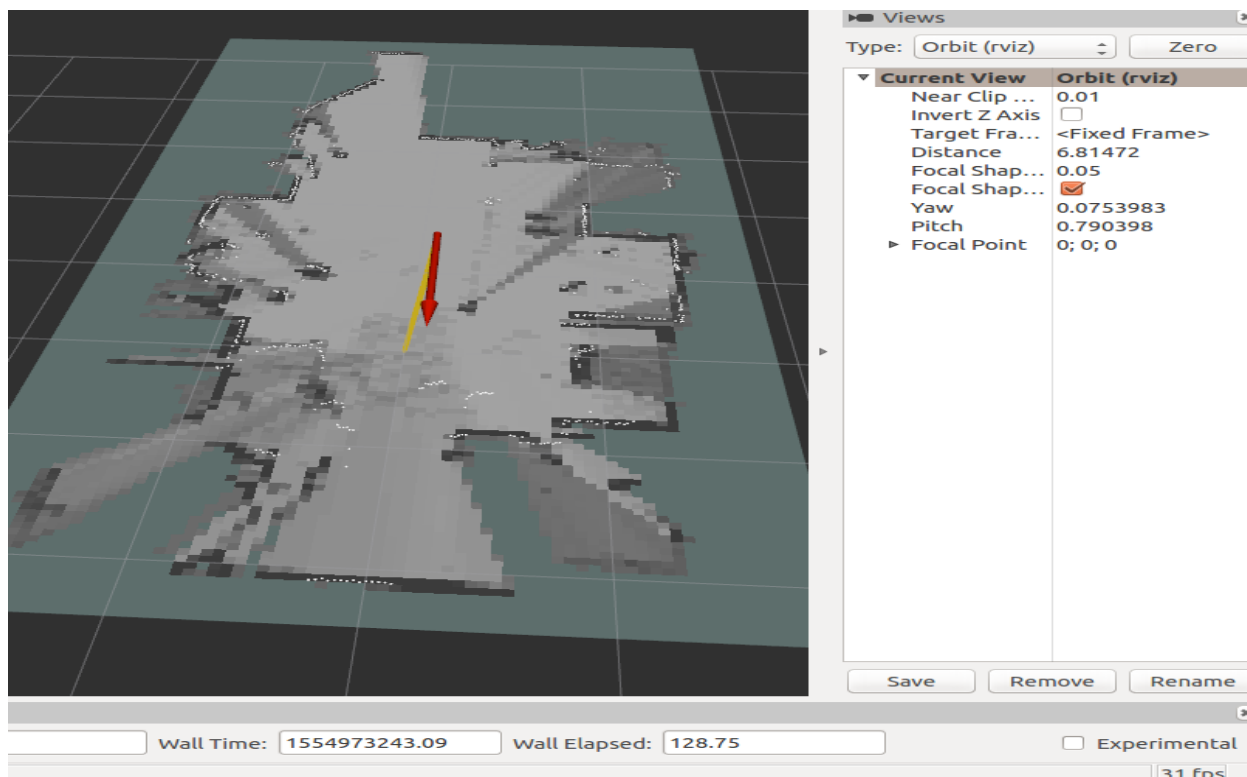


Рисунок 9.3 – Результат формирования 2D карты в пакете RViz

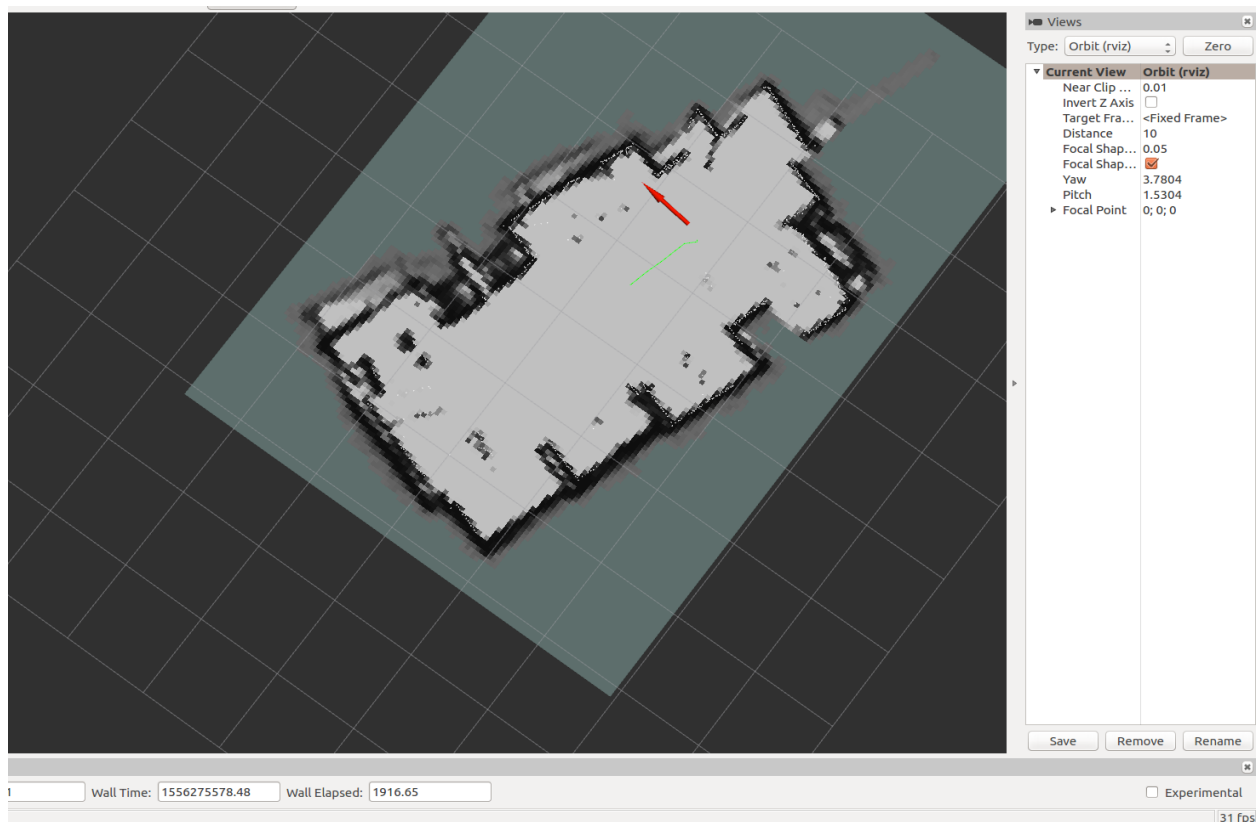


Рисунок 9.4 – Результат формирования 2D карты в пакете RViz

Карта формируется в режиме реального времени. На формируемом участке карты белым цветом выделяются свободные области, не содержащие объектов-препятствий для продвижения мобильной навигационной платформы. Черным цветом выделяются области содержащие препятствия, мешающие дальнейшему движению. Полученная таким образом карта, в дальнейшем используется для мобильных систем с автономной навигацией по заранее известной местности.

В конечном результате был собран макет прототипа мобильной навигационной платформы. Собранный макет прототипа представлен на рисунках 9.5 - 9.6.



Рисунок 9.5 – Макет прототипа мобильной навигационной платформы



Рисунок 9.6 – Макет прототипа мобильной навигационной платформы

Макет представляет собой платформу размером 800 х 500 х 300. Внутри платформы установлены 2 мотора-редуктора, пусковое устройства и аккумулятор. На переднюю часть платформы установлен датчик Velodyne VLP16. В центральной части платформы размещается портативная ЭВМ – Jetson AGV Xavier.

ЗАКЛЮЧЕНИЕ

В магистерской диссертации были изучены и освещены различные SLAM-алгоритмы программного комплекса пространственной навигации и мониторинга, фундаментальные принципы и методы управления и локализации в задачах синтеза мобильных роботов. Проведён детальный анализ и описание концепции и инструментов фреймворка ROS.

Была разработана система автоматизированного управления мобильной навигационной платформой. Особенностью разработанной САУ является использование в качестве основного и единственного датчика типа лидар – Velodyne VLP16. Разработанные алгоритмы подсистемы навигации позволяют формировать как 2D, так и 3D карты местности. Полученные карты, в дальнейшем могут быть использованы в мобильных системах автономной навигации.

Были выполнены следующие разделы задания:

- знакомство с алгоритмами SLAM-навигации программного комплекса пространственной навигации и мониторинга;
- изучение методов управления и локализации в задачах синтеза мобильных роботов
- изучение средств фреймворка ROS по работе со SLAM алгоритмами;
- разработка АСУ мобильным роботом
- выбор структуры автоматической системы регулирования;
- разработка структурной схемы подсистемы навигации;
- разработка подсистемы телеуправления;
- реализация и сравнение работы алгоритмов SLAM-навигации основанные на использовании лазерных дальномеров;
- интеграция фреймворка Google Cartographer с алгоритмами Obstacle Avoidance.

В основном, все алгоритмы SLAM состоят из инициализации, оценки движения камеры, оценки 3D-структуры, глобальной оптимизации и релокализации. Для повышения эффективности работы изученных методов можно прибегнуть к объединению визуальных и инерциальных данных, мы можем получить более стабильные результаты оценки. Кроме того, возможно использование детальной информации о датчике для решения оценки масштаба и компенсации искажения входной информации. В настоящее время практически все смартфоны и планшетные устройства имеют камеры, GPS, гироскоп и акселерометр. Таким образом, сопряжение датчиков - одно из направлений для реализации надежных и практичных систем автоматического регулирования для современных мобильных роботов, базирующихся на алгоритмах SLAM-навигации.

Материалы исследований были опубликованы в статьях и материалах конференции [18, 54-58].

СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ

1. Chatila R, Laumond JP (1985) Position referencing and consistent world modeling for mobile robots. In: Proceedings of International Conference on Robotics and Automation Vol. 2. pp 138–145.
2. Durrant-Whyte H, Bailey T (2006) Simultaneous localization and mapping: part i. Robot Autom Mag IEEE 13(2):99–110
3. Bailey T, Durrant-Whyte H (2006) Simultaneous localization and mapping (slam): Part ii. IEEE Robot Autom Mag 13(3):108–117.
4. Thrun S, Leonard JJ (2008) Handbook of robotics. Chap. Simultaneous localization and mapping
5. Aulinas J, Petillot YR, Salvi J, Lladó X (2008) The slam problem: a survey. In: Proceedings of Conference on Artificial Intelligence Research and Development: Proceedings of International Conference of the Catalan Association for Artificial Intelligence. pp 363–371.
6. Billinghurst M, Clark A, Lee G (2015) A survey of augmented reality. Found Trends Human-Computer Interact 8(2-3):73–272.
7. Engel J, Sturm J, Cremers D (2012) Camera-based navigation of a low-cost quadrocopter. In: Proceedings of International Conference on Intelligent Robots and Systems. pp 2815–2821.
8. Ros G, Sappa A, Ponsa D, Lopez AM (2012) Visual slam for driverless cars: a brief survey. In: Intelligent Vehicles Symposium (IV) Workshops.
9. Fuentes-Pacheco J, Ruiz-Ascencio J, Rendón-Mancha JM (2015) Visual simultaneous localization and mapping: a survey. Artif Intell Rev 43(1):55–81.
10. Klette R, Koschan A, Schluns K (1998) Computer vision: three-dimensional data from images. 1st edn.
11. Nister D (2004) A minimal solution to the generalised 3-point pose problem. In: Proceedings of IEEE Conference on Computer Vision and Pattern Recognition Vol. 1. pp 560–5671.

12. Grisetti G, Kümmerle R, Stachniss C, Burgard W (2010) A tutorial on graph-based slam. *Intell Transp Syst Mag IEEE* 2(4):31–43.
13. Kümmerle R, Grisetti G, Strasdat H, Konolige K, Burgard W (2011) g2o: A general framework for graph optimization. In: *Proceedings of International Conference on Robotics and Automation*. pp 3607–3613.
14. Bundle adjustment a modern synthesis. In: Triggs B, McLauchlan PF, Hartley RI, Fitzgibbon AW (eds) (2000) *Vision algorithms: theory and practice*. pp 298–372.
15. Klein G, Murray DW (2007) Parallel tracking and mapping for small AR workspaces. In: *Proceedings of International Symposium on Mixed and Augmented Reality*. pp 225–234.
16. Nistér D, Naroditsky O, Bergen J (2004) Visual odometry. In: *Proceedings of IEEE Conference on Computer Vision and Pattern Recognition Vol. 1*. p 652.
17. Yousif K, Bab-Hadiashar A, Hoseinnezhad R (2015) An overview to visual odometry and visual slam: applications to mobile robotics. *Intell Ind Syst* 1(4):289–311.
18. Пославский С.И., Окунский М.В. Построение карт для мобильного робототехнического объекта при помощи визуальной одометрии // *Когнитивная робототехника: материалы II международной конференции / под ред. В.И. Сырямкина, А.В. Юрченко; Национальный исследовательский Томский государственный университет*, с. 28.
19. Fraundorfer F, Scaramuzza D (2012) Visual odometry: Part ii: matching, robustness, optimization, and applications. *Robot Autom Mag IEEE* 19(2):78–90.
20. Engel J, Sturm J, Cremers D (2013) Semi-dense visual odometry for a monocular camera. In: *Proceedings of International Conference on Computer Vision*. pp 1449–1456.

21. Engel J, Schöps T, Cremers D (2014) LSD-SLAM: large-scale direct monocular SLAM. In: Proceedings of European Conference on Computer Vision. pp 834–849.
22. Kerl C, Sturm J, Cremers D (2013) Robust odometry estimation for RGB-D cameras. In: Proceedings of International Conference on Robotics and Automation. pp 3748–3754.
23. Kerl C, Sturm J, Cremers D (2013) Dense visual SLAM for RGB-D cameras. In: Proceedings of International Conference on Intelligent Robots and Systems. pp 2100–2106.
24. Agarwal S, Furukawa Y, Snavely N, Simon I, Curless B, Seitz SM, Szeliski R (2011) Building rome in a day. *Commun ACM* 54(10):105–112.
25. Civera J, Grasa OG, Davison AJ, Montiel J (2010) 1-point ransac for extended kalman filtering: application to real-time structure from motion and visual odometry. *J Field Robot* 27(5):609–631.
26. Davison AJ (2003) Real-time simultaneous localisation and mapping with a single camera. In: Proceedings of International Conference on Computer Vision. pp 1403–1410.
27. Davison AJ, Reid ID, Molton ND, Stasse O (2007) Monoslam: real-time single camera SLAM. *Pattern Anal Mach Intell IEEE Trans* 29(6):1052–1067.
28. Nistér D (2004) An efficient solution to the five-point relative pose problem. *Pattern Anal Mach Intell IEEE Trans* 26(6):756–770.
29. Williams B, Klein G, Reid I (2007) Real-time SLAM relocalisation. In: Proceedings of International Conference on Computer Vision. pp 1–8.
30. Castle R, Klein G, Murray DW (2008) Video-rate localization in multiple maps for wearable augmented reality. In: 2008 12th IEEE International Symposium on Wearable Computers. pp 15–22. doi:10.1109/ISWC.2008.4911577.
31. Klein G, Murray DW (2009) Parallel tracking and mapping on a camera phone. In: Proceedings of International Symposium on Mixed and Augmented Reality. pp 83–86.

32. Newcombe RA, Davison AJ (2010) Live dense reconstruction with a single moving camera. In: Proceedings of IEEE Conference on Computer Vision and Pattern Recognition. pp 1498–1505.
33. Pradeep V, Rhemann C, Izadi S, Zach C, Bleyer M, Bathiche S (2013) MonoFusion: real-time 3D reconstruction of small scenes with a single web camera. In: Proceedings of International Symposium on Mixed and Augmented Reality. pp 83–88.
34. Strasdat H, Montiel JM, Davison AJ (2012) Visual SLAM: why filter? *Image Vision Comput* 30(2):65–77.
35. Mei C, Sibley G, Cummins M, Newman P, Reid I (2009) A constant-time efficient stereo slam system. In: Proceedings of British Machine Vision Conference. pp 54–15411.
36. Cummins M, Newman P (2008) FAB-MAP: probabilistic localization and mapping in the space of appearance. *Int J Robot Res* 27(6):647–665
37. Strasdat H, Montiel J, Davison AJ (2010) Scale drift-aware large scale monocular slam. In: Proceedings of Robotics: Science and Systems. p 5.
38. Mur-Artal R, Montiel JMM, Tardós JD (2015) ORB-SLAM: a versatile and accurate monocular SLAM system. *IEEE Trans Robot* 31(5):1147–1163. doi:10.1109/TRO.2015.2463671.
39. Mur-Artal R, Tardós JD (2016) ORB-SLAM2: an open-source SLAM system for monocular, stereo and RGB-D cameras. *CoRR*. abs/1610.06475.
40. Eade E, Drummond T (2009) Edge landmarks in monocular slam. *Image Vis Comput* 27(5):588–596.
41. Hirose K, Saito H (2012) Fast line description for line-based SLAM. In: Proceedings of the British Machine Vision Conference.
42. Klein G, Murray D (2008) Improving the agility of keyframe-based SLAM. In: Proceedings of European Conference on Computer Vision. pp 802–815.

43. Newcombe RA, Lovegrove SJ, Davison AJ (2011) DTAM: dense tracking and mapping in real-time. In: Proceedings of International Conference on Computer Vision. pp 2320–2327.
44. Okutomi M, Kanade T (1993) A multiple-baseline stereo. *Pattern Anal Mach Intell IEEE Trans* 15(4):353–363.
45. Rudin LI, Osher S, Fatemi E (1992) Nonlinear total variation based noise removal algorithms. *Phys D Nonlinear Phenom* 60(1):259–268.
46. Ondruska P, Kohli P, Izadi S (2015) MobileFusion: real-time volumetric surface reconstruction and dense tracking on mobile phones. *IEEE Trans Vis Comput Graph* 21(11):1251–1258.
47. Stühmer J, Gumhold S, Cremers D (2010) Real-time dense geometry from a handheld camera. In: Goesele M, Roth S, Kuijper A, Schiele B, Schindler K (eds). Springer, Berlin, Heidelberg, pp 11–20.
48. Schöps T, Engel J, Cremers D (2014) Semi-dense visual odometry for AR on a smartphone. In: Proceedings of International Symposium on Mixed and Augmented Reality. pp 145–150.
49. Caruso D, Engel J, Cremers D (2015) Large-scale direct SLAM for omnidirectional cameras. In: Proceedings of International Conference on Intelligent Robots and Systems.
50. Engel J, Stueckler J, Cremers D (2015) Large-scale direct SLAM with stereo cameras. In: Proceedings of International Conference on Intelligent Robots and Systems.
51. Forster C, Pizzoli M, Scaramuzza D (2014) SVO: fast semi-direct monocular visual odometry. In: Proceedings of International Conference on Robotics and Automation. pp 15–22.
52. Baker S, Matthews I (2004) Lucas-kanade 20 years on: a unifying framework. *Int J Comput Vis* 56(3):221–255.
53. Engel J, Koltun V, Cremers D Direct sparse odometry. *CoRR*. abs/1607.02565.

54. Пославский С.И., Окунский М.В. Методы локализации мобильных робототехнических систем // Когнитивная робототехника: материалы II международной конференции / под ред. В.И. Сырямкина, А.В. Юрченко; Национальный исследовательский Томский государственный университет, с. 29

55. Пославский С.И., Окунский М.В. Алгоритмы последовательной локализации робототехнических систем на плоскости // Когнитивная робототехника: материалы II международной конференции / под ред. В.И. Сырямкина, А.В. Юрченко; Национальный исследовательский Томский государственный университет, с. 45-50.

56. Пославский С. И. Методы управления движением мобильного робота // Инноватика-2018. Сборник материалов XIV Международной школы-конференции студентов, аспирантов и молодых ученых. г. Томск, с. 111-113.

57. Пославский С. И. Основные концепции фреймворка ROS // Инноватика-2018. Сборник материалов XIV Международной школы-конференции студентов, аспирантов и молодых ученых 26–27 апреля 2018 г. г. Томск, с. 114-116.

58. Пославский С. И. Архитектура SLAM навигации программного продукта Octomap // Инноватика-2018. Сборник материалов XIV Международной школы-конференции студентов, аспирантов и молодых ученых. г. Томск, с. 117-120.



АНТИПЛАГИАТ
ТВОРИТЕ СОБСТВЕННЫМ УМОМ

Томский Государственный
Университет

СПРАВКА

о результатах проверки текстового документа на наличие заимствований

Проверка выполнена в системе
Антиплагиат.ВУЗ

Автор работы	Пославский Сергей
Подразделение	ФИТ, кафедра информационного обеспечения инновационной деятельности
Тип работы	Магистерская диссертация
Название работы	АЛГОРИТМЫ SLAM-НАВИГАЦИИ В ЗАДАЧАХ СИНТЕЗА АВТОНОМНЫХ МОБИЛЬНЫХ РОБОТОВ
Название файла	2019_Poslavskiy_FIT.pdf
Процент заимствования	5,98%
Процент цитирования	10,96%
Процент оригинальности	83,06%
Дата проверки	09:21:18 18 июня 2019г.
Модули поиска	Кольцо вузов; Модуль поиска "ТГУ"; Модуль поиска общеупотребительных выражений; Коллекция Патенты; Модуль поиска перефразирований Интернет; Модуль поиска перефразирований eLIBRARY.RU; Коллекция Медицина; Модуль поиска Интернет; Коллекция ГАРАНТ; Коллекция eLIBRARY.RU; Модуль поиска переводных заимствований; Цитирование; Коллекция РГБ; Сводная коллекция ЭБС; Модуль выделения библиографических записей; Модуль поиска ИПС "Адилет"
Работу проверил	Гольцова Полина Андреевна ФИО проверяющего
Дата подписи	18.06.19. Подпись проверяющего

Чтобы убедиться
в подлинности справки,
используйте QR-код, который
содержит ссылку на отчет.



Ответ на вопрос, является ли обнаруженное заимствование
корректным, система оставляет на усмотрение проверяющего.
Предоставленная информация не подлежит использованию
в коммерческих целях.