



МИНОБРНАУКИ РОССИИ
Федеральное государственное бюджетное образовательное учреждение высшего образования
«МИРЭА – Российский технологический университет»

РТУ МИРЭА

Институт искусственного
интеллекта (ИИИ)
Кафедра проблем управления (КПУ)

РАБОТА ДОПУЩЕНА К ЗАЩИТЕ

Заведующий кафедрой

Подпись

М.П. Романов

«16» 06 2022 г.

ВЫПУСКНАЯ КВАЛИФИКАЦИОННАЯ РАБОТА

по направлению подготовки бакалавров
15.03.06 «Мехатроника и робототехника»

На тему: Навигационная система мобильного робота на основе технического зрения

Обучающийся:

подпись

Санин Никита Максимович

Ф.И.О.

Шифр

21K8912

Группа

КРБО-02-18

Руководитель
выпускной работы

подпись

К.т.н., доцент

ученая степень, ученое звание, должность

Кучерский Р.В.

Ф.И.О.

Консультант
выпускной работы

подпись

Ассистент

ученая степень, ученое звание, должность

Голубов В.В.

Ф.И.О.

Консультант по
экономической
части

подпись

К.э.н., доцент

ученая степень, ученое звание, должность

Бондалетова Н.Ф.

Ф.И.О.

Москва 2022 г.



МИНОБРНАУКИ РОССИИ
Федеральное государственное бюджетное образовательное учреждение высшего образования
«МИРЭА – Российский технологический университет»
РТУ МИРЭА

Институт искусственного интеллекта
Кафедра проблем управления

СОГЛАСОВАНО

Заведующий Кафедрой
М.П. Романов

Подпись

« 20 » апреля 2022 г.

УТВЕРЖДАЮ

Директор института
М.П. Романов

Подпись

« 20 » апреля 2022 г.

ЗАДАНИЕ

на выполнение выпускной квалификационной работы бакалавра

Обучающийся: Санин Никита Максимович

Шифр: 21K8912

Направление подготовки: 15.03.06 «Мехатроника и робототехника»

Группа: КРБО-02-18

1. Тема выпускной квалификационной работы

Навигационная система мобильного робота на основе технического зрения.

2. Техническое задание на выполнение выпускной квалификационной работы

2.1. Цель:

2.1.1. Повышение эффективности навигационной системы мобильного робота

2.2. Задачи, требующие решения для достижения поставленной цели:

2.2.1. Анализ проблем и обзор существующих систем навигации мобильных роботов

2.2.2. Обзор и анализ систем навигации на базе лидара и на базе RGB камеры

2.2.3. Обзор и анализ алгоритмов локальной навигации и картографирования monoSLAM

2.2.4. Разработка макета программного обеспечения на основе предложенных алгоритмов

2.2.5. Проведение комплексных экспериментальных исследований по оценке работоспособности созданного макета программного обеспечения

2.3. Основные требования к результату разработки

2.3.1. Разработанный макет программного обеспечения должен обеспечивать решения задач локальной навигации и картографирования местности автономным мобильным роботом

2.3.2. Качественное соответствие реальной карты местности и построенной роботом карты и качественная оценка полноты построения карты

2.4. Планируемые результаты выполнения выпускной квалификационной работы

2.4.1. Обзор и анализ существующих навигационных систем мобильных роботов

2.4.2. Обзор и анализ систем навигации на базе лидара и на базе RGB камеры

2.4.3. Обзор и анализ алгоритмов локальной навигации и картографирования monoSLAM

2.4.4. Созданный макет программного обеспечения для выполнения лабораторных работ

2.4.5. Пояснительная записка, описывающая особенности разработки

3. Этапы выполнения выпускной квалификационной работы

| № этапа | Содержание этапа выпускной квалификационной работы | Результат выполнения этапа ВКР | Срок выполнения |
|---------|--|---|-----------------|
| 1 | Анализ проблем и обзор существующих навигационных систем мобильных роботов | Выбор навигационной системы мобильных роботов | 20.04.22 |
| 2 | Обзор и анализ систем навигации на базе лидара и на базе RGB камеры | Выбор навигационной системы на базе RGB камеры | 13.05.22 |
| 3 | Обзор и анализ алгоритмов локальной навигации и картографирования monoSLAM | Выбор алгоритма monoSLAM для локальной навигации и картографирования | 21.05.22 |
| 4 | Разработка макета программного обеспечения на основе предложенных алгоритмов | Макет разработанного программного обеспечения | 22.05.22 |
| 5 | Проведение комплексных экспериментальных исследований по оценке работоспособности созданного макета программного обеспечения | Результаты комплексных экспериментов, подтверждающие работоспособность созданного макета программного обеспечения | 25.05.22 |



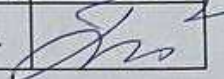
4. Содержание экономического раздела

- 4.1. Разработка основных разделов бизнес-плана проекта: анализ рынка сбыта, конкурентоспособности и маркетинга.
- 4.2. Организация и планирование работ по теме.
- 4.3. Определение договорной цены.
- 4.4. Оценка экономической целесообразности проведения работ по теме.

5. Перечень разрабатываемых документов и графических материалов

- 5.1. Блок-схема макета разработанного программного обеспечения.
- 5.2. Технические характеристики камеры мобильного робота, используемой в экспериментальном исследовании.
- 5.3. Блок-схема алгоритма локальной навигации и картографирования.
- 5.4. Технические характеристики мобильного робота, используемого в экспериментальном исследовании.
- 5.5. Результат сравнительного анализа алгоритмов локальной навигации и картографирования.
- 5.6. Результаты работоспособности макета разработанного программного обеспечения.

6. Руководитель и консультанты выпускной квалификационной работы

| Функциональные обязанности | Должность в Университете | Ф.И.О. | Подпись |
|---|-----------------------------|------------------|---|
| Руководитель выпускной работы | Доцент | Кучерский Р.В. |  |
| Консультант выпускной работы | Ассистент | Голубов В.В. |  |
| Консультант по экономическому разделу выпускной работы | Доцент, к.э.н. | Бондалетова Н.Ф. |  |

Задание выдал

Задание получил к выполнению

Руководитель ВКР  Кучерский Р.В.

Обучающийся  Санин Н.М.

«20» апреля 2022 г.

«20» апреля 2022 г.

Оглавление

| | |
|--|-----------|
| Аннотация..... | 0 |
| Введение | 4 |
| 1 Анализ проблем и обзор существующих систем навигации мобильного робота | 5 |
| 1.1 Анализ проблем навигационной системы мобильного робота | 5 |
| 1.2 Обзор существующих систем навигации мобильного робота..... | 7 |
| 1.2.1 Инерциальная система навигации..... | 7 |
| 1.2.2 Система технического зрения..... | 8 |
| Заключение к 1 главе | 11 |
| 2 Обзор и анализ систем навигации на базе лидара и на базе RGB камеры..... | 12 |
| 2.1 Алгоритм одновременной навигации и картографирования SLAM... 12 | |
| 2.2 Сенсоры для получения информации о местоположении | 13 |
| 2.3 Обзор навигационных систем на базе лидара | 15 |
| 2.3.1 Алгоритм Gmapping..... | 15 |
| 2.3.2 Алгоритм Hector-SLAM | 16 |
| 2.3.3 Алгоритм Google Cartographer..... | 17 |
| 2.3.4 Алгоритм RTAB-Map | 19 |
| 2.4 Обзор навигационных систем на базе RGB камеры..... | 20 |
| 2.4.1 Алгоритм LSD-SLAM..... | 20 |
| 2.4.2 Алгоритм ORBSLAM | 21 |
| Заключение ко 2 главе | 22 |
| 3 Обзор и анализ алгоритмов локальной навигации и картографирования monoSLAM..... | 23 |
| 3.1 Фильтр Калмана..... | 23 |
| 3.2 Обзор и анализ алгоритма ORBSLAM..... | 27 |
| 3.2.1 Дескриптор особых точек ORB | 27 |
| 3.2.2 Принцип работы алгоритма ORBSLAM..... | 29 |
| 3.2.3 Замыкание циклов и ключевые кадры SLAM..... | 33 |
| Заключение к 3 главе | 40 |

| | |
|--|-----------|
| 4 Разработка макета программного обеспечения на основе алгоритма ORBSLAM и проведение комплексных экспериментальных исследований по оценке работоспособности созданного макета программного обеспечения | 41 |
| 4.1 Мобильный автономный робот NVIDIA JetBot..... | 41 |
| 4.2 Среда симулятора Gazebo | 44 |
| 4.3 Камера мобильного робота NVIDIA JetBot..... | 45 |
| 4.4 Создание макета программного обеспечения | 47 |
| 4.5 Проведение экспериментальных исследований | 48 |
| 4.6 Критерии качества карты местности, построенной с помощью алгоритма | 53 |
| Заключение к 4 главе | 55 |
| 5 Организационно-экономическая часть | 56 |
| 5.1 Введение..... | 56 |
| 5.2 Разработка основных разделов бизнес-плана проекта..... | 56 |
| 5.2.1 Описание продукта | 56 |
| 5.2.2 Анализ рынка сбыта..... | 57 |
| 5.2.3 Конкурентоспособность | 57 |
| 5.2.4 План маркетинга | 58 |
| 5.3 Организация и планирование работ по теме | 58 |
| 5.3.1 Организация работ | 59 |
| 5.3.2 График проведения работ..... | 62 |
| 5.3 Определение договорной цены | 62 |
| 5.4 Оценка экономической целесообразности проведения работ | 67 |
| Заключение к 5 главе | 68 |
| Список литературы | 69 |
| Приложение | 72 |

Аннотация

В состав пояснительной записки входит: 22 рисунка, 60 страницы, 3 приложения, 23 источника.

Тема выпускной квалификационной работы: «Навигационная система мобильного робота на основе технического зрения». В процессе работы был произведён обзор и анализ методов локальной навигации на основе технического зрения робота, а также обзор и анализ методов решения задачи monoSLAM применительно на мобильном роботе NVIDIA JetBot.

Разработка велась с использованием языка программирования Python 3.6.5 в среде симулятора Gazebo, который является частью ROS (Robot Operation System) – функциональная среда программирования роботов.

Введение

В настоящее время в опасных для жизни и здоровья человека сферах деятельности (поисковые и спасательные работы, транспортировка медикаментов в опасных зонах, разминирование и т.д.) существует потребность в замене человека на автономного мобильного робота, в связи с этим возникает и актуальная проблема - необходимость точного определения параметров собственного местоположения автономного мобильного робота и самостоятельного построения карт местности. Существуют инерциальные системы навигации и системы навигации, основанные на применении GPS, но основным недостатком данных систем навигации является невозможность обеспечения необходимой точности определения текущего местоположения автономного мобильного робота, поэтому становится востребованной возможность развития технологий локальной навигации на основе визуальной обратной связи с применением системы технического зрения. Такая навигационная система позволяет с высокой точностью осуществить оценку не только координат самого робота, но и окружающих его объектов. Картографирование местности автономными мобильными роботами также является насущной проблемой, возможность решения которой зависит от качества информационно-измерительных средств, а также от параметров окружающей среды. Обе ранее упомянутые проблемы решает алгоритм SLAM.

1 Анализ проблем и обзор существующих систем навигации мобильного робота

1.1 Анализ проблем навигационной системы мобильного робота

Основной проблемой современной робототехники является решение вопроса навигационных систем. Для успешного решения задачи навигации бортовой компьютер робота должен грамотно решать такие задачи, как построение маршрута, управление параметрами движения (задавать скорость вращения колес и угол их поворота), правильная интерпретация сведений об окружающей среде, получаемых от датчиков, и постоянный мониторинг собственных координат.

Глобальная спутниковая система GPS, ежегодно набирает популярность у разработчиков систем навигации автономных роботов. На сегодняшний день существует много аналогов GPS-системы, но их применение в глобальной навигации пока ограничено задачами соблюдения правильности общего курса, так как это связано с основным требованием законов робототехники о точности навигации — ошибка в определении собственных координат не может превышать размера самого автономного робота (в противном случае возникают неполадки или конфликты со средой). Поэтому в качестве основной навигация GPS применяется в основном в автопилотах крупных океанских лайнеров или гражданских и военных самолетов[18]. Кроме того, в различных регионах Земли, на местности со сложным рельефом, естественными помехами и в зданиях GPS-сигнал может приниматься с помехами и неустойчиво. Таким образом, данная система еще продолжительный отрезок времени не сможет использоваться как основная в задачах глобальной навигации небольших аппаратов.

В вопросе локальной навигации присутствуют следующие варианты решения вышеупомянутых проблем: идея использования для ориентирования на местности искусственных сооружений (например, специальных вышек, «маяков»). Данная идея в основном реализована в коммерческих версиях, и аппарат, оснащенный системой технического зрения, может довольно точно рассчитать расстояние до вышки-метки благодаря анализу изменения геометрических размеров в зависимости изменения зоны видимости. Если же нет возможности установить искусственные маяки, робот может попытаться самостоятельно выделить некоторые статичные объекты окружающей среды (например, высокое дерево, гора) и выполнять привязку своих координат к этим объектам. Но основной недостаток такого подхода – проблема зависимости от параметров окружающей среды (например, уровня освещенности). Возможное решение - использование стереокамер, зная угол зрения каждой камеры, возможно вычислить расстояние до искомой цели. Но такое внедрение значительно скажется на итоговой стоимости автономной машины. Другая возможность реализации локальной навигации — использование радиомаяков. В зоне действий робота размещаются источники радиосигналов, которые обрабатываются бортовым компьютером автономного робота. Но такой подход также не лишен недостатков: так как радиомаяки расположены в определенных точках некоторого маршрута, робот не сможет выбирать альтернативный путь движения и теряет возможность обходить препятствия. В связи с чем встает вопрос о создании адаптивной локальной навигационной системы автономного мобильного робота, которая будет подстраиваться под особенности внешней среды функционирования робота.

1.2 Обзор существующих систем навигации мобильного робота

1.2.1 Инерциальная система навигации

Инерциальная система навигации — метод, в основе которого показания с датчиков, таких как акселерометров, датчики поворота, гироскопов и т.д. По показанию данных приборов определяются координаты положения объекта. Данная навигационная система была популярна в недалёком прошлом, так как была устойчива к помехам и позволяла автоматизировать процесс передвижения[18].

Инерциальные навигационные системы (ИНС) в основном представлены сенсорами линейного ускорения (акселерометрами) и угловой скорости (гироскопами или парой акселерометров для измерения центробежного ускорения). Гироскопы способствуют измерению усилия (момент внешней силы), прикладываемого к телу, на котором они размещены, и на этой основе определять положение тела относительно позиции, с которой началось движение, и определяют скорость. Механические акселерометры похожим методом определяют собственное ускорение. Основное уравнение для вычисления координат имеет вид:

$$\bar{\omega} = \frac{\partial r_m}{\partial t^2} - \bar{F}(r_m),$$

где ω — измеряемое акселерометром ускорение, r_m — радиус-вектор центра тяжести воспринимающего объекта в инерциальной системе, F — ускорение свободного падения, t — время.

Недостатком данной системы является накопление ошибки определения положения в пространстве за время работы, то есть чем дольше объект, оборудованный ИНС, движется, тем больше значение погрешности в

определении координат положения. Свойства окружающей среды также оказывают свое влияние на показания: неоднородная поверхность передвижения, проскальзывание, дрейф, всё это сказывается на показаниях датчиков, поэтому полноценно на них не стоит полагаться. Однако существуют комплексированные системы, учитывающие показания нескольких отдельных датчиков. Например, это GPS и данные с акселерометра, в таком случае часто используется метод Калмана или фильтр частиц. Основная проблема заключается в том, что сигнал GPS не всегда одинаково достигаем, а зачастую и вовсе пропадает в местах появления помех: под землей, под толстым слоем бетонных стен (подвальные помещения).

1.2.2 Система технического зрения

Основное функциональное предназначение системы технического зрения заключается в том, чтобы мобильный робот мог обработать изображение местности, полученное с камеры, и распознать или же определить на ней свойства объектов: их вид, форму или другие атрибуты и, имея представление благодаря полученным данным, выполнять поставленные цели.

Система технического зрения даёт более точное представление данных, поскольку в отличие от инерциальных, не возникает особо крупных накопленных ошибок, которые бы оказывали сильное влияние на качество выполняемого процесса. В основном, система технического зрения представлена такими приборами, как лидары, камеры, радары, эхолоты, стереокамеры, камеры глубины и т.д. Основными задачами для данных систем являются:

- 1) Получение изображения: изображения получают с одного или нескольких датчиков изображения, включая датчики расстояния,

ультразвуковые камеры и т. д. Полученные изображения могут быть двухмерными или трехмерными.

2) Предварительная обработка: прежде чем к видеоданным можно будет применить методы компьютерного зрения для извлечения определенного объема информации, видеоданные должны быть обработаны таким образом, чтобы они удовлетворяли определенным условиям, в зависимости от используемого метода.

3) Нахождение и разбиение: на конкретном этапе разработки проходит процесс стратификации на важные точки, ключевые единицы..

4) Высокоуровневая обработка: на данном этапе полученные входные данные представляют собой небольшой набор данных, к примеру, область изображения или набор точек, которые должны содержать определенный объект.

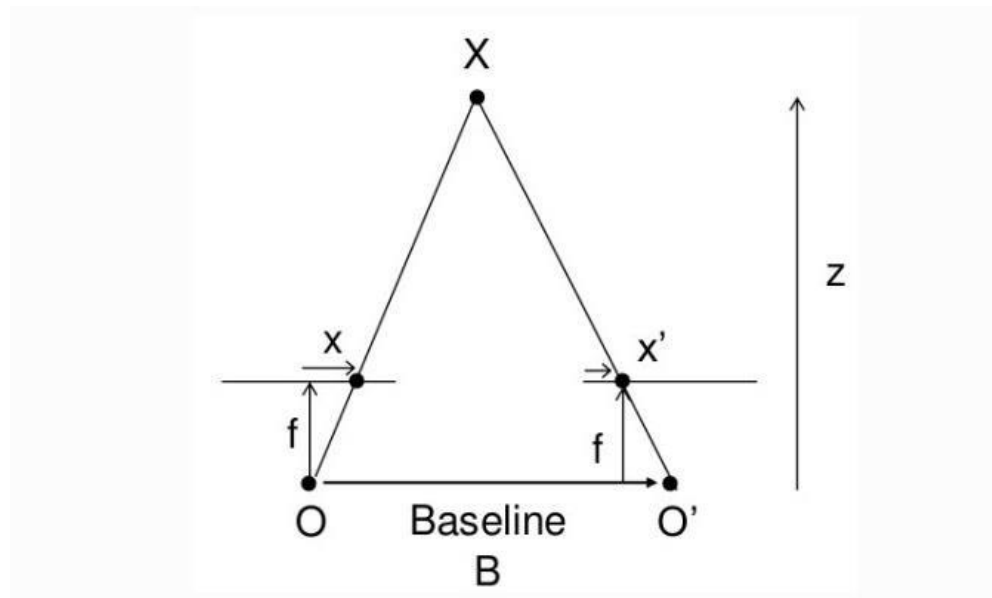


Рисунок 1 Принцип определения расстояния до объектов

Несколько (двух более чем достаточно) стереокамер на борту робота способны получить карту глубины, в которой матрица изображения имеет 3 координаты: x , y , z , где координаты x , y - это номер столбца и строки пикселя, соответственно, а третья координата z - это рассчитанная по формуле 3 координата, показывающая уровень глубины.

$$disparity = x - \bar{x} = \frac{Bf}{z}$$

Если речь идёт о видеопотоке, то нужно преобразование последовательности изображений в облако точек для получения более-менее конкретного представления о положении объектов в пространстве относительно робота. Пример приведён на рисунке 2.

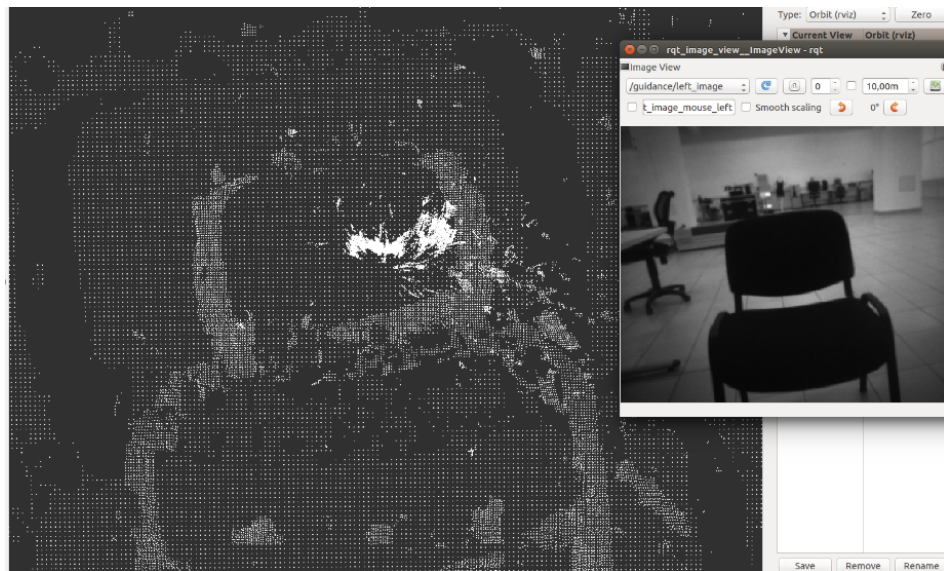


Рисунок 2 Карта глубины из тестового датасета

Облако точек аналогично можно получить, используя светочувствительные измерительные приборы, таких как лидар, например (рисунок 3). Система лазеров способна передавать роботу информацию о расстоянии до объектов в обход использования карты глубины. Принцип работы такой системы состоит в том, что лазер функционирует как сонар на подводной лодке, испускает лучи света, которые отражаются от поверхности ближайшего объекта и возвращаются обратно, где фоточувствительный элемент принимает их и проводится расчёт времени реакции датчика, после вычисляется расстояние, учитывая, что скорость луча равна скорости света.

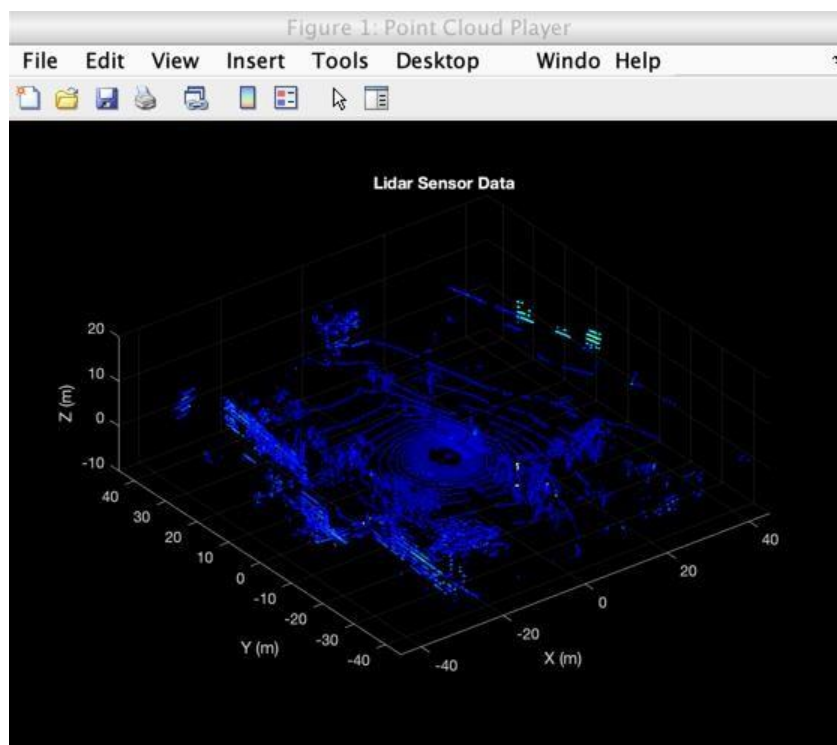


Рисунок 3 Облако точек, полученное благодаря использованию лидара

Заключение к 1 главе

В первой главе был проведен обзор и анализ проблем навигационных систем как глобального уровня (GPS), так и локального. Проблема реализации локальной навигационной системы заключается в невозможности системы адаптироваться под изменяющиеся параметры окружающей среды.

Мобильный робот может ориентироваться только в пределах изученной местности, оборудованной особыми метками. Также был проведен обзор существующих методов решения вопроса навигации автономным роботом, за исключением использования алгоритма SLAM, так как он будет рассмотрен в следующей главе.

2 Обзор и анализ систем навигации на базе лидара и на базе RGB камеры

2.1 Алгоритм одновременной навигации и картографирования SLAM

Как следует из заключения к 1 главе, основной проблемой реализации локальной навигационной системы является невозможности системы адаптироваться под изменяющиеся параметры окружающей среды. Данную задачу может решить алгоритм одновременной локализации и картографирования SLAM (англ. SLAM – simultaneous localization and mapping). Данный метод используется в мобильных автономных аппаратах для построения карты в неизвестном пространстве или для обновления карты в заранее известном пространстве с одновременным контролем текущего местоположения и пройденного пути. Навигация и построение карты являются напрямую зависят друг от друга и выполняются совместно – для записи новых данных и проверки уже записанных ранее объектов окружения (препятствий, стен и пр.) мобильному роботу необходимо сопоставлять координаты расположения объектов с собственными координатами, а для определения собственных координат робот должен использовать уже известную информацию о расположении окружающих объектов. Используя методы одновременной навигации и картографирования автономный мобильный как подсистему системы управления, автономный робот может самостоятельно ориентироваться в пространстве и формировать модель окружения. Среди областей применения SLAM можно выделить обход зданий в экстремальных условиях, например, в целях разминирования, или инспекцию потенциально опасных объектов, к примеру электростанции или морские порты. Алгоритм вариативен и может быть применим с разными сенсорами и датчиками, которыми оборудован автономный робот для получения информации о местоположении.

2.2 Сенсоры для получения информации о местоположении

Для того, чтобы система локальной навигации работала, автономный мобильный робот должен быть оборудован какой-нибудь сенсорной входной измерительной системой, способной определять местоположение относительно ориентиров. Самыми распространенными датчиками можно считать акустические, лазерные и оптические. У каждого типа датчика имеются свои плюсы и минусы использования, ошибки, возникающие во время эксплуатации, и специфика использования, такая, как и энергетические и материальные расходы. Поэтому следует проанализировать свойства датчиков и определить наиболее подходящий для решения поставленной задачи.

Акустические сенсоры (эхолокаторы, сонары и т.д.) довольно популярны благодаря их точности, простоте и низкому энергопотреблению (0.01-1 Вт), низкой стоимости и скорости вычислений. Сенсор передает звуковую волну определенной частоты, она отражается от препятствия и передается обратно на сенсор. Расстояние до препятствия робот рассчитывает по времени между отправкой сигнала и принятием отраженного сигнала. При этом дальность, на которой акустические сенсоры работают успешно, ограничена и составляет для разных датчиков от 2 до 10 метров.

Системы лазерной локации (лидары) – это точные активные датчики. Они часто используются в мобильных и воздушных автономных роботах благодаря простоте вычислений, высокой дальности и обнаружению препятствий сразу на всех направлениях. Принцип работы схож с эхолокационными датчиками, только вместо звуковой волны используется световой луч: лазер испускает лучи света, которые отражаются от поверхности ближайшего объекта и возвращаются обратно, где фоточувствительный элемент принимает их и

проводится расчёт времени реакции датчика, после вычисляется расстояние, учитывая, что скорость луча равна скорости света. Лидары более точны, чем оптические датчики, особенно в условиях дождя и тумана, дальность их использования варьируется от 20 до 100 м, при этом они высоко энергозатратны (50-200 Вт).

Оптические датчики зрения различаются довольно сильно в зависимости от использованной технологии:

Монокулярные камеры (RGB) просты, экономичны и занимают мало места. При этом вычисления расстояний до объектов достаточно ресурсозатратны. Для упрощения вычислений можно использовать стереокамеру. Для решения задачи навигации используется фильтр на основе признаков (feature) в комбинации с фильтром Калмана, также это решает задачу loop closure – узнавания роботом уже посещенного места и обновления карты в соответствии с этими данными. Вычисления также получаются достаточно сложными, так как вектор состояния значительно увеличивается при исследовании большого пространства. Энергопотребление для моно- и стереокамер составляет 0.01-10 Вт и 2-15 Вт, соответственно.

RGB-D камера (Microsoft Kinect sensor) это один из наиболее широко используемых в SLAM сенсоров, так как она сочетает в себе монокулярную камеру и камеру глубины. Она удобна для построения 3D-карты и обладает средней экономической эффективностью. RGB-D камера работает по методу структурированного света и времяпролетной технологии (англ. Time-of-flight camera). Структурированный свет работает по следующему принципу: проектор излучает световые узоры (паттерны), которые деформируются при отражении от объекта. Затем одна или несколько камер распознают 3D-геометрию с помощью алгоритмов триангуляции. Времяпролетная технология используется для создания изображений, которые в качестве пикселей содержат оценки расстояний от экрана до конкретных точек наблюдения. TOF берет начало от алгоритмов, используемых в радарах. Благодаря этому

формируется дальностное изображение, подобное радиолокационным портретам, за исключением того, что для его построения задействован световой импульс вместо радиочастотного сигнала. RGB-D камера дешевле лидара, но дороже обычной монокулярной камеры. Энергопотребление составляет от 2 до 5 Вт.

При тщательном сравнении монокулярной (RGB) камеры и лидара, становится понятно, что экономически выгоднее использование обычной камеры, так как она сама по себе дешевле и обладает меньшим энергопотреблением, но также присутствуют и минусы эксплуатации, среди которых зависимость от освещенности окружающей среды и меньшая зона видимости камеры по сравнению с лидаром. Проведем обзор вариаций реализации навигационных систем на базе лидара и на базе RGB моно-камеры.

2.3 Обзор навигационных систем на базе лидара

2.3.1 Алгоритм Gmapping

Алгоритм Gmapping является реализацией алгоритма одновременной локализации и картографирования с использованием лидара. Принцип работы алгоритма следующий: алгоритм создает карту благодаря полученным данным с лидара. В основном используется 2D лидар, способный сканировать окружающее пространство на 360 градусов вокруг. Далее, происходит сопоставление и сравнение готовой карты и данных, полученных с лидара, в результате чего происходит оценка и корректировка местоположение робота. Плюс использования данного алгоритма заключается в том, что в режиме реального времени можно построить локальную карту высокой точности, но это применимо только для небольших

локаций, так как чем больше изучаемая локация, тем больше становится объем памяти и вычислений для построения карты местности.

Картирование основано на фильтре частиц Рао–Блэквелла, который оценивает апостериорное значение вероятности:

$$p(x_{1:t}|z_{1:t}, u_{0:t})$$

где $x_{1:t}$ - потенциальная траектория робота, используя наблюдения $z_{1:t}$ и данные одометрии $u_{0:t}$. Апостериорное значение вероятности аппроксимируется множеством точек (частиц) с соответствующими вероятностями (весами). Частица с максимальным весом рассматривается как реальное состояние мира. Вес частицы обновляется мерой соответствия между новым сканером и картой, оцененной с помощью сопоставителя сканов. GMapping использует метод градиентного спуска для оценивания соответствия сканированиям. На каждой итерации проверяется несколько predetermined направлений. В качестве начальной позиции для следующей итерации выбирается позиция вдоль направления с максимальной совпадающей оценкой[5]. Оценка сканирования рассчитывается следующим образом:

$$score(scan, map) = \sum_{p \in scan} e^{\frac{-1}{\sigma} d(p, map)^2}$$

где p – точка скана, $d(p, map)$ – минимальное между точкой скана и препятствием на карте, σ – предустановленное значение дисперсии.

2.3.2 Алгоритм Hector-SLAM

Алгоритм использует данные, полученные с лидара, и инерциальной навигационной системы, формируя таким образом двухмерную модель окружающей местности. Данный алгоритм не требует значительной

вычислительной мощности и может использоваться при выравнивании небольших полигонов, а также в сценариях, где необходима высокая скорость обновления данных. Алгоритм требует использования высокоточного лидара, если же при построении используется лидар с низкой точностью воспроизведения, то это приведет к невозможности построения карты местности.

Принцип работы: предполагая, что положение робота известно, после каждого лазерного сканирования данных вероятность занятости реальными препятствиями на карте сетки продолжает увеличиваться и в конечном итоге бесконечно приближается к 1, а место без препятствий приближается к 0. Вся карта делится на подсетки с определенным разрешением. Распределение вероятностей финальной карты выглядит следующим образом:

$$p(m_i|z_{1:t}, x_{1:t}) = f(p(m_i|z_{1:t-1}, x_{1:t-1}), p(m_i|z_t, x_t))$$

2.3.3 Алгоритм Google Cartographer

Алгоритм Google Cartographer можно представить как две отдельные, но связанные между собой системы: глобальный SLAM и локальный SLAM. Карта окружающей среды хранится в виде графа, где каждая вершина представляет собой дополнительную подкарту и сканы, полученные после создания соответствующей подкарты. Локальный SLAM используется для создания подкарт региона. Глобальный SLAM используется для того, чтобы связать подкарты вместе настолько последовательно насколько это возможно. Ребра представляют собой преобразования между соответствующими подчиненными картами. У локального SLAM для каждой подкарты появляется накапливающаяся ошибка. В следствие с этим помимо процесса сопоставления сканирования в глобальном SLAM появляется шаг оптимизации, чтобы сделать собранную карту согласованной и попытаться

избавиться от полученной ошибки. Процесс построения подкарты итеративный и состоит из повторяющегося согласования кадров с камеры (сканов) и кадров координат подкарты (фреймов). Несколько идущих друг за другом сканов используются для построения подкарты в форме сетки вероятности, где каждой точке размера $r*r$ присваивается определенная вероятность от p_{min} до p_{max} .

$$odds(p) = \frac{p}{1-p}$$

$$M_H(cell) = odds^{-1}(odds(M_{CT}(cell) * odds(p_n)))$$

где M_{CT} – предыдущая вероятность клетки, p_n – вероятность попадания, $odds(p)$ – шанс события, где p – искомая вероятность того, что событие произойдет, $(1-p)$ – вероятность того, что событие не произойдет.

Основная идея подхода к сопоставлению сканов в том, чтобы минимизировать значение функционала. Таким образом, процесс сканирования совпадений сводится к минимизации следующего функционала:

$$arg_{\varepsilon} min \sum_{K=1}^K (1 - M_{ГЛАД}(T_{\varepsilon} h_k))^2$$

где $M_{глад}(x)$ – значение ячейки, сглаженное значениями по соседству, h_k – клетка, хранящая точку текущего скана, T_{ε} – промежуточная матрица, сдвигающая точку h_k на ε , ε – вектор смещения, имеющий вид $(\xi x, \xi y, \xi \theta)^T$

К тому же, в сравнении с алгоритмом GMapping, Cartographer может быть реализован одновременно с несколькими лидарами (например, при создании многоуровневой карты местности), а также с стерео- и камерами глубины для получения большего объема информации об окружающей среде. В общем, Cartographer – это удобный в плане настройки вариант реализации

алгоритма SLAM. Он более требовательный к вычислительным мощностям, чем GMapping, но позволяет получить более качественный результат.

2.3.4 Алгоритм RTAB-Map

RTAB-Map является подходом SLGB RGB-D Graph, основанный на теореме Байеса детектор замыкания контура, то есть на цикле прогнозирования-коррекции. Для определения вероятности появления нового изображения из предыдущего или нового местоположения детектор использует подход с набором слов. В график карты добавляется новое ограничение после принятия гипотезы о замыкании цикла, а затем запускается процедура минимизации ошибки в карте при помощи оптимизатора графика. Источником данных для карты чаще всего являются такие сенсоры как лидар, стерео- и камеры глубины RGB-D, но в основном как источник данных используется RGB-D камера[9].

Алгоритм использует дескриптор особых точек SURF, который по 2 входным изображениям ищет ключевые точки. Затем алгоритм приступает к процедуре сопоставления и сравнения изображения с массивом данных, содержащим сделанные ранее ключевые изображения (key frames). Необходимо нахождение вероятности совпадения изображения, полученного через камеру, с изображениями, сохраненными ранее в массиве данных, используя Баясовский фильтр. Для определения ориентации объектов в пространстве относительно сцены необходимо использование одометрических датчиков, расположенных на колёсах робота, или других способов регистрации дальнометрических данных, для картографирования местности, параллельно с этим применяя показания датчиков для использования метода замкнутых контуров.

2.4 Обзор навигационных систем на базе RGB камеры

Существует большое количество различных вариаций реализации алгоритма одновременной локализации и картографирования SLAM, но в основном большинство алгоритмов использует камеру RGB-D как источник получения информации о местоположении, поэтому уместно рассмотреть варианты, когда реализуется именно монокулярный SLAM, так называемый *monoSLAM*.

2.4.1 Алгоритм LSD-SLAM

LSD-SLAM[5] это современный монокулярный алгоритм для локальной навигации и картографирования. Вместо того чтобы извлекать особые точки этот алгоритм работает с непосредственным изображением. Три одновременных процесса для формирования ключевых кадров. Исходя из плотности отдельных областей на различных участках карты глубины изображения получается геометрия тел на сцене. Для получения карты глубины необходима фильтрация множества взаимосвязанных кадров, которые расположены на конкретном пикселе, попарно образующем стерео-изображения в сравнении с каждым отдельно взятым пикселем. После построенный граф видимости трех позиций позволяет построить крупные карты, устойчивые к ошибкам масштабирования, для учета возможности замыкания циклов (*loop closure*). Преимуществом LSD-SLAM является прямой подход к Visual SLAM, так как в нём содержится вся информация, полученная из изображения, включая края и границы предметов, а алгоритмы, основанные на детектировании ключевых точек, обходят стороной очертания и границы и ориентированы только на распределение самих ключевых точек[14]. Исходя из этого такой алгоритм обладает

большей надежностью и точностью в среде с внешними атрибутами предметов и однообразной текстурой. Множество сравниваемых стереопар, заменяющие нескольких объёмных кадров являются частью фильтров глубины, ориентированных на показания камеры относительно одного пикселя.

Хотя и LSD-SLAM и является довольно практичным и точным алгоритмом, его программное обеспечение очень ресурсозатратно, так как использует большой объем вычислительных мощностей, к тому же его ПО не предусматривает подключение и передачу данных через другие платформы, к примеру, такие как ROS, поэтому этот алгоритм не подойдет для поставленной задачи.

2.4.2 Алгоритм ORBSLAM

ORBSLAM основан на монокулярной системе SLAM, которая работает в режиме реального времени, в пространстве любых масштабов[2]. Система предоставляет надёжную информацию о положении объектов в пространстве, так как устойчива при хаотичном движении, к ротации, и инициирует замыкание циклов траектории движения. Она включает в себя полную автоматическую инициализацию. ORBSLAM строит не очень плотную карту, так как ключевые точки находятся в относительно большой дистанции друг от друга. Система предназначена для реализации основных модулей SLAM: перемещение, замыкание цикла, отслеживание и отображение. В заключение получается отслеживаемая и компактная, растущая только при наличии содержимого сцены карта на выходе. В сравнении с другими алгоритмами, система крайне производительна. Данный алгоритм может быть использован на работе с монокулярной камерой и с RGB-D камерой, а также поддерживается использование стерео-камер. Также данный алгоритм весьма

оптимизирован для использования мощностей бортового компьютера, например, такого мобильного робота, как NVIDIA JetBot. Более подробный обзор и описание принципа работы данного алгоритма будет в 3 главе.

Заключение ко 2 главе

Во второй главе был проведен обзор систем навигации на основе алгоритма одновременной локализации и картографирования SLAM на базе лидара и на базе монокулярной RGB камеры (без модуля камеры глубины). Также были проведены краткие обзоры датчики, благодаря которым поступают входные данные для мобильного робота, из заключения которых следует, что технология лидара значительно дороже и затратнее в обслуживании, нежели RGB камера. Из экономических соображений, гораздо выгоднее реализовать навигационную систему на базе монокулярной камеры, но проблема заключается в том, что недостающий модуль глубины придется компенсировать посредством добавления дополнительных расчетных операций, что сопутствует увеличению вычислительной мощности, как в случае алгоритма LSD-SLAM.

Алгоритм LSD-SLAM является хорошим вариантом для реализации навигационной системы, однако он не поддерживает внешний источник поступления видеоданных, и соответственно его невозможно применить в связке с таким программным обеспечением как ROS.

Из рассмотренных алгоритмов выбор останавливается на ORBSLAM. Этот алгоритм ничуть не уступает по возможностям его конкурентам, к тому же алгоритм совместим с ROS.

3 Обзор и анализ алгоритмов локальной навигации и картографирования **monoSLAM**

Во второй главе были рассмотрены 2 варианта реализации монокулярного SLAM: LSD-SLAM и ORBSLAM. Было принято решение использовать алгоритм ORBSLAM для подробного обзора и анализа и последующего использования для создания макета программного обеспечения. В задаче SLAM необходимо использование фильтров для получения точных, непрерывно обновляемых оценок положения в реальном времени и для уточнения положения пространственных ориентиров в окружающей среде.

3.1 Фильтр Калмана

Функция фильтра Калмана заключается в получении оценки о состоянии объекта в пространстве относительно его предыдущего положения. Состояние фильтра задается двумя переменными: x_k – вектор состояния системы во время k , полученный по результату наблюдений до момента k включительно; p_k – ковариационная матрица ошибок (мера точности оценивания вектора состояния) в момент времени k , задающая оценку точности полученной оценки вектора состояния и включающая в себя ковариации, показывающие выявленные взаимосвязи между параметрами состояния системы, и оценку дисперсий погрешности вычисленного состояния[22]. Итерации фильтра Калмана делятся на две фазы: экстраполяция и коррекция. Экстраполяция (предсказание) вектора состояние системы по оценке вектора состояния и примененному вектору управления с шага $(k - 1)$ на шаг k :

$$\hat{x}_{k|k-1} = F_k \hat{x}_{k-1|k-1} + B_k u_{k-1}$$

Ковариационная матрица для экстраполированного вектора состояния:

$$P_{k|k-1} = F_k P_{k-1|k-1} F_k^T + Q_{k-1}$$

Во время этапа коррекции рассчитывается отклонение полученного на шаге k наблюдения от наблюдения, ожидаемого при произведенной экстраполяции:

$$\tilde{y}_k = z_k - H_k \hat{x}_{k|k-1}$$

Ковариационная матрица для вектора отклонения (вектора ошибки):

$$S_k = H_k P_{k|k-1} H_k^T + R_k$$

Оптимальная по Калману матрица коэффициентов усиления, формирующаяся на основании ковариационных матриц, имеющейся экстраполяции вектора состояния и полученных измерений (посредством ковариационной матрицы вектора отклонения)[19]:

$$K_k = P_{k|k-1} H_k^T S_k^{-1}$$

Коррекция ранее полученной экстраполяции вектора состояния – получение оценки вектора состояния системы:

$$\hat{x}_{k|k} = \hat{x}_{k|k-1} + K_k \tilde{y}_k$$

Расчет ковариационной матрицы оценки вектора состояния системы:

$$P_{k|k} = (I - K_k H_k) P_{k|k-1}$$

Только при использовании приведенного оптимального вектора коэффициентов выражение для ковариационной матрицы оценки вектора состояния системы справедливо. В общем случае это выражение имеет более сложный вид. Несмотря на то, что фильтр Калмана применим только к линейным системам, и не подходит для коррекции положения мобильного автономного робота, он эффективен и вполне подходит для оценок положения неподвижных пространственных точек.

Расширенный фильтр Калмана (ЕКФ) – это функционально тот же фильтр Калмана, только который можно использовать ещё и в нелинейных процессах. ЕКФ является одним из наиболее популярных вариантов решения задачи SLAM. Он позволяет уточнять положение всех обнаруженных ориентиров и провести оценку положения робота на карте.

В расширенном фильтре Калмана (ЕКФ) состояние системы и наблюдения не обязаны быть линейными функциями состояния, а должны быть дифференцируемыми:

$$x_k = f(x_{k-1}, u_{k-1}) + w_{k-1}$$

$$z_k = h(x_k) + v_k$$

Приближение и наблюдение с использованием первых производных функций состояния является основным фактором применения расширенного фильтра Калмана.

Экстраполяция (предсказание) вектора состояние системы по оценке вектора состояния и примененному вектору управления с шага $(k - 1)$ на шаг k :

$$\hat{x}_{k|k-1} = f(\hat{x}_{k-1|k-1}, u_{k-1})$$

Ковариационная матрица для экстраполированного вектора состояния:

$$P_{k|k-1} = F_{k-1}P_{k-1|k-1}F_{k-1}^T + Q_{k-1}$$

При этапе коррекции отклонение полученного на шаге k наблюдения от наблюдения, ожидаемого при произведенной экстраполяции:

$$\tilde{y}_k = z_k - h(\hat{x}_{k|k-1})$$

Ковариационная матрица для вектора отклонения (вектора ошибки):

$$S_k = H_k P_{k|k-1} H_k^T + R_k$$

Оптимальная по Калману матрица коэффициентов усиления, формирующаяся на основании ковариационных матриц, имеющейся экстраполяции вектора состояния и полученных измерений (посредством ковариационной матрицы вектора отклонения):

$$K_k = P_{k|k-1} H_k^T S_k^{-1}$$

Коррекция ранее полученной экстраполяции вектора состояния – получение оценки вектора состояния системы:

$$\hat{x}_{k|k} = \hat{x}_{k|k-1} + K_k \tilde{y}_k$$

Расчет ковариационной матрицы оценки вектора состояния системы:

$$P_{k|k} = (I - K_k H_k) P_{k|k-1}$$

Матрицы изменения состояния системы и наблюдения определяются Якобианами:

$$F_{k-1} = \frac{\partial f}{\partial x} \big|_{\hat{x}_{k-1|k-1}, u_{k-1}}$$

$$H_k = \frac{\partial h}{\partial x} \big|_{\hat{x}_{k|k-1}}$$

К сожалению, расширенный фильтр Калмана не лишен недостатков, среди которых тот факт, что фильтр не является оптимальным, в отличие от линейного аналога. Также, в случае если процесс смоделирован некорректно или начальное вычисление состояния системы ошибочно, результаты могут быстро расходиться из-за линеаризации.

3.2 Обзор и анализ алгоритма ORBSLAM

При применении монокулярной камеры возникает необходимость решения задачи поиска соответствующих точек на изображениях, полученных с камеры. Данная задача состоит из двух частей: поиск ключевых точек (key point) и нахождение соответствий между ними. К найденным точкам выдвигаются дополнительные требования помимо поиска соответствий на двух изображениях: устойчивость к повороту изображений, растяжению, к изменению точки наблюдения, к появлению шума. Так как монокулярный алгоритм ORBSLAM использует одну обычную RGB камеру, то полученное изображение с камеры робота конвертируется в черно-белый формат для поиска особых точек посредством дескриптора ORB.

3.2.1 Дескриптор особых точек ORB

Алгоритм ORB (Oriented FAST and Rotated BRIEF) представляет собой улучшенную версию детектора ключевых точек FAST посредством слияния с дескриптором BRIEF с некоторыми модификациями[10]. Изначально для определения ключевых точек используется FAST. Алгоритм FAST быстро сравнивает яркость пикселя s с окружающими 16 пикселями, которые находятся в маленьком круге вокруг s . После сравнения пиксели в круге сортируются по трем классам (светлее s , темнее s или похожие на s)(рисунок 4).

$$I_p > I_c + t$$

$$I_p < I_c - t$$

$$I_c - t < I_p < I_c + t$$

где I – яркость пикселей, t – некоторый заранее фиксированный порог по яркости, p – проверяемый пиксель, кандидат на особую точку.

Если более 8 пикселей темнее или ярче s , то кандидат p выбирается в качестве ключевой точки. Таким образом, ключевые точки, найденные с помощью fast, дают нам информацию о расположении определяющих краев на изображении.

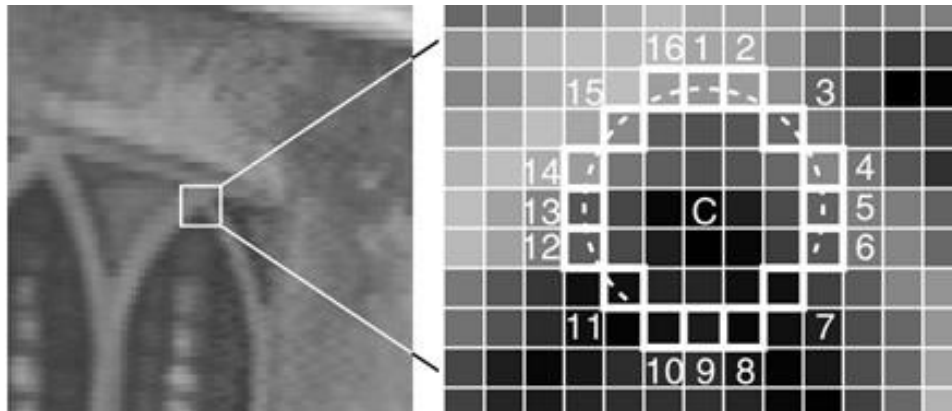


Рисунок 4 Реализация алгоритма FAST для обнаружения ключевых точек

После того, как особые точки найдены, вычисляют их наборы признаков (дескрипторы), характеризующие окрестность каждой особой точки. BRIEF преобразует их в двоичный вектор признаков, чтобы вместе они могли представлять объект. Вектор двоичных признаков, также известный как дескриптор бинарных признаков, представляет собой вектор признаков, который содержит только 1 и 0. Вкратце, каждая ключевая точка описывается вектором признаков, который представляет собой строку из 128–512 битов. Бинарный тест между точками x и y определяется так:

$$\tau(P, x, y) = \begin{cases} 1: p(x) < p(y) \\ 0: p(x) \geq p(y) \end{cases}$$

Алгоритм ORB добавляет устойчивости к ротации для дескриптора BRIEF, так как дескриптор BRIEF плохо работает при вращении в плоскости. Для этого необходимо вычислить угол ротации, сначала вычисляются моменты яркости для окрестности особой точки:

$$m_{pq} = \sum_{x,y} x^p y^q I(x, y)$$

где x, y – пиксельные координаты, I – яркость. И затем угол ориентации особой точки:

$$\theta = \text{atan2}(m_{01}, m_{10})$$

Направление вектора от этой угловой точки к центру задает ориентацию. Имея угол ориентации особой точки, последовательность точек для бинарных сравнений в дескрипторе BRIEF поворачивается в соответствие с этим углом. Новые положения для точек бинарных тестов вычисляются так:

$$\begin{pmatrix} x_i' \\ y_i' \end{pmatrix} = R(\theta) * \begin{pmatrix} x_i \\ y_i \end{pmatrix}$$

По полученным точкам вычисляется бинарный дескриптор BRIEF. После нужно отсортировать “правильные” тесты[7].

3.2.2 Принцип работы алгоритма ORBSLAM

Очень сложно визуализировать ситуацию вокруг камеры, одновременно отслеживая траекторию фокусировки. Алгоритм SLAM показал, что такая задача является чрезмерно дорогостоящей со стороны вычислительных ресурсов. Необходимы следующие инструкции для получения работоспособных результатов:

А) Выбор среди подмножества экспериментальных кадров ключевых точек, полученных благодаря наблюдению за особенностями окружающей сцены;

Б) Отсутствие повторного отбора ключевых точек, обусловленного увеличением их количества;

В) Набор ключевых точек и кадров должен максимально вариативен, то есть кадры и точки них должны отличаться от соседних;

Г) Оптимизация параметров карты и ее масштабируемости при построении локальной карты местности;

Д) Так как выбрана замкнутая циклическая траектория робота, то необходима возможность быстрого замыкания циклов траектории движения в режиме реального времени.

Первоначальным источником идеи замкнутых траекторных циклов в ключевых точках изображений с камеры были Мюррей и Кляйн с проектом параллельного отслеживания и картографирования окружающей среды (Parallel Tracking and Mapping) PTAM[1]. Данный алгоритм, несмотря на ограничения для мелкомасштабных операций, предоставляет простые, но в то же время довольно эффективные способы детектирования особых точек, триангуляции точек, сопоставления объектов, локализации камеры для каждого участка карты и перераспределения после возникновения ошибки отслеживания. К сожалению, некоторые факторы серьезно ограничивают его использование: отсутствие замыкания цикла и низкая инвариантность к точке перемещения, а также необходимость вмешательства человека для начальной загрузки карты. ORBSLAM основан на работе PTAM и вносит следующие улучшения в свою работу:

А) Использование одних и тех же ключевых точек для всех задач: отслеживание, отображение, перемещение и замыкание цикла. Это делает систему более эффективной и надежной. Поиск особых точек в режиме реального времени выполняется благодаря функции дескриптора ORB, способствуя устойчивости к изменениям ракурса и освещению.

Б) Работа в реальном времени в больших пространствах. Картографирование и отслеживание сосредоточены в локальной видимой области, благодаря использованию графика видимости, независимо от глобальных размеров карты.

В) Заккрытие цикла в реальном времени на основе оптимизации графа, который выполняет функцию основного графа.

Г) Основанная на выборе модели, позволяющей воссоздание сцены любого вида новая автоматическая и надежная процедура инициализации.

Чтобы выполнить перестановку камеры и замыкание цикла, нам нужен метод распознавания изначального положения. Предпочтительнее, чтобы распознаватель был как можно более полным, например, он всё равно смог бы распознать своё окружение, при изменении ракурса и положения камеры. Был выбран очень эффективный распознаватель DBoW2, основное преимущество которого учет масштабирования особых точек дескриптора ORB.

Обзор принципа работы DBoW2: для определения соответствия изображению повторного места, необходимо воспользоваться методом Bag of Words[4], который по найденным признакам изображения, полученными с помощью дескриптора ORB, составляет визуальные слова и суммирует их, получая вектор признаков, отличающийся для каждого изображения по наличию находящихся на нем ключевых точек. Эти визуальные слова в итоге составляют элементы визуального словаря. Словарь, созданный в автономном режиме при помощи DBoW2, структурирован в виде дерева больших наборов дескрипторов, извлеченных из набора данных обучающих изображений. Извлечение ключевых точек и их дескрипторов, которые назначаются визуальному слову, проходящему через словарь, является основным критерием обработки нового изображения. Расстояния вычисляются по расстоянию Хэмминга, так как дескрипторы представляют собой бинарный код. В результате полученный вектор слов для каждого присутствующего на изображении слова содержит термин «частота инверсии» (tf-idf). Этот показатель тем выше, чем чаще встречается слово на изображении, и тем меньше он был в наборе обучающих данных. Далее необходимо сравнение полученного вектора с групповым вектором слов изображений в базе данных с использованием обратного индекса, сохраняющегося для каждого слова, в каких изображениях оно появилось.

Сходство между двумя векторами слов v_1 и v_2 вычисляется по формуле:

$$s(v_1, v_2) = 1 - \frac{1}{2} \left| \frac{v_1}{|v_1|} - \frac{v_2}{|v_2|} \right|$$

Необходимо сравнение полученной оценки с оценкой ожидания для изображения, указывающей на то же место. Она соотносится с предыдущим кадром, при условии, что обрабатывается последовательность изображений перекрывающимися друг друга.

$$n(v_i, v_j) = \frac{s(v_i, v_j)}{s(v_i, v_j - 1)}$$

Кандидат цикла чтобы не быть отклоненным, проходит процедуру согласования с предыдущими кадрами. Это означает, что перекрывающуюся последовательность с одним из k изображений образуют группы с самыми высокими показателями для последних данных изображений. Надежность системы повышается благодаря проверки согласованности

После успешного прохождения всех проверок кандидат в цикле принимается. DBoW2 вычисляет фундаментальную матрицу с помощью RANSAC, поскольку изначально применялся к видеопотокам без всякой обработки (без 3-мерных данных). В случае если особые точки принадлежат одному и тому же узлу на уровне дерева словаря, то поиск начальных соответствий между этими точками выполняется полностью. Прямой индекс, который использует база данных, сохраняет для каждой функции в изображении принадлежащий ему узел.

Основой проекта является DBoW2[4] с функциональным применением ORB для распознавания мест дескриптором с различных ракурсов. Используется реализация ORB в библиотеке OpenCV.

Параметры по умолчанию этой реализации состоят в том, чтобы извлечь 500 особых точек в 8 различных масштабах с масштабным коэффициентом 1,2. Поскольку особые точки выделяются в нескольких масштабах для одной и той же позиции изображения, они плохо распределены на изображении. По мере

того, как проходит этап подавления немаксимумов, эффективно улучшается распределение особых точек, но значительно снижает производительность сопоставления дескрипторов[12]. Вместо этого мы извлекаем 1000 ключевых точек, что улучшает их распределение и отклик нашего распознавателя за счет небольшого увеличения времени извлечения ORB.

3.2.3 Замыкание циклов и ключевые кадры SLAM

А) Набор ключевых кадров

При поступлении нового ключевого кадра необходимо применение дескриптора ORB для связки визуального слова с каждым дескриптором, пересекающим дерево словаря. Затем нужно составить BoW вектор и вставить его в базу данных кадров ключевых. Необходимо применить только дескриптор ORB для детекции ключевых точек вместо использования функций предоставляемых PTAM для гарантии производительности распознавателя[11]. Недостовверные результаты, предоставленные описанием ключевых точек PTAM, полученных с помощью ORB, получаются из-за пирамиды масштабов, которую PTAM использует для построения графов, со значением коэффициента два.

Б) Поиск кандидата на завершение цикла

После завершения обработки новых ключевых кадров отделом программы, отвечающим за картографирование, и выполнения локальной настройки, извлекается первый ключевой кадр в очереди цикла, содержащий только последнее изображение. После необходим поиск средством распознавания мест в базе данных ключевых кадров кандидата в цикле, при условии соответствия трем предыдущим совпадениям ключевых кадров. В случае нахождения кандидата, необходима проверка на закрытость цикла PTAM, при этом выполняя поиск общих точек измерения карты между

текущим и ключевым кадром-кандидатом цикла, и если цикл замкнут, то отпадает необходимость дополнительных вычислений.

В) Геометрическая проверка: преобразование подобия

Вычисление преобразования из текущей системы координат камеры ключевого кадра в систему кандидата в цикл является первым этапом корректировки цикла. Единственным способом узнать лишнее смещение, которое накопилось в ходе экспериментальных исследований, которое в monoSLAM может иметь семь степеней свободы: три перемещения, три поворота и масштаб, и является данное преобразование. Именно для этого нужно вычисление преобразования подобия, $\text{Sim}(3)$, из текущего кандидата цикла l ключевого кадра:

$$S_{k,l} = \begin{bmatrix} S_{k,l} R_{k,l} & t_{k,l} \\ 0 & 1 \end{bmatrix}$$

Где $s \in R^+$ это коэффициент масштабирования, $R \in SO(3)$ - матрица вращения, а $t \in R^3$ - вектор перемещения. Для принятия решения о начале исправления цикла (если успешно пройдено) или отвержению кандидата на цикл (в противном случае) необходима геометрическая проверка посредством вычисления этого преобразования[5].

Необходим поиск набора стартовых соответствий между объектами ORB: текущем и ключевом кадре-кандидате на цикл для вычисления преобразование.

Функции ORB, используемые для распознавания, отличаются от функций, используемых PTAM для отображения. Интерполируется глубина каждого объекта с его тремя ближайшими отслеживаемыми объектами PTAM, чтобы восстановить трехмерные координаты объектов ORB.

Теперь, используя схему RANSAC, необходимо найти преобразование подобия, поддерживаемое достаточным количеством соответствий. На каждой итерации выбор трех соответствия и вычисление преобразования подобия. Затем, проверяя ошибку сопоставления трехмерных точек на обоих

изображениях, идет подсчет, сколько соответствий поддерживают это преобразование. В случае, если сумма ошибки сопоставления пикселей в текущем и циклическом изображениях-кандидатах меньше 6 пикселей, то соответствие считается верным. Геометрическая проверка считается успешной, в случае если RANSAC находит преобразование, которое поддерживает 40% исходных соответствий менее чем за 70 итераций. Уточнение преобразования посредством повтора его снова со всеми значениями, и повторное вычисление, в случае нахождения дополнительных значений.

Г) Улучшение цикла

Модуль настройки реализует процесс оптимизации позиции и точки на всей карте, чтобы результативно замкнуть цикл. Однако из-за накопленного смещения текущее состояние карты далеко от верного. По этой причине, оптимизируя позиционный граф, сформированный из текущей позиции в ключевой кадр цикла, вычисляется решение для начального положения[1].

Сначала необходимо при поддержке вращения и перемещения конвертирование для всех позиций их SE (3) абсолютное преобразование, $T_{i,w}$, в подобие $\text{Sim}(3)$, $S_{i,w}$, устанавливая масштаб на 1. После этого необходим расчет относительного преобразования $\Delta S_{i,j}$, между двумя позициями: текущей и следующей, при этом параллельно проходит закрытие цикла с преобразованием подобия, вычисленным между текущим ключевым кадром и ключевым кадром цикла $S_{k,l}$.

После этого расчет минимизации остаточной ошибку $r_{i,j}$ между положением $S_{i,w}$, и $S_{j,w}$, относительно ограничения $\Delta S_{i,j}$ в касательном пространстве $\text{sim}(3)$ и в минимальном представлении:

$$r_{i,j} = (\log_{\text{sim}(3)}(\Delta S_{i,j} * S_{j,w} * S_{i,w}^{-1}))_{\text{sim}(3)}^v$$

Функция для минимизации ошибки выглядит так:

$$x^2 = \sum_{i,j} r_{i,j}^T \delta_{i,j} r_{i,j}$$

Где $\delta_{i,j}$ - обратная ковариация остаточного $r_{i,j}$ и устанавливается в единицу. Необходимо исключить позицию ключевого кадра цикла из оптимизации, чтобы зафиксировать семь степеней свободы решения. Используя метод Левенберга - Марквардта, реализованный в оптимизаторе графов g2o, для использования факта разбросанности ошибки с помощью метода CHOLMOD, основанного на разложении Холецкого.

После того, как положения камеры были оптимизированы, необходимо обновить 3D точки, связанные с ними. Для каждой точки x_j выбираем исходный кадр $T_{i,w}$ и добавляем точку на карту, используя формулу:

$$x_j^{cor} (S_{i,w}^{cor})^{-1} * T_{i,w} * x_j$$

Финальный этап – это преобразование исправленного преобразования подобия $cor S_{i,w}$ обратно в 3D трансформацию объекта $cor T_{i,w}$. Не столь важно подобие масштаба, так как масштаб точки был обновлен формулой ранее, и сохранение угла поворота, так как она не теперь не зависит от масштаба, но при это все равно необходимо выполнение перемасштабирования.

$$S_{i,w}^{cor} = \begin{bmatrix} sR & t \\ 0 & 1 \end{bmatrix} \rightarrow T_{i,w}^{cor} = \begin{bmatrix} R & \frac{1}{s}t \\ 0 & 1 \end{bmatrix}$$

Положение и точки будут оптимизированы исходя из их взаимных соединений, когда РТАМ будет выполнять процедуру глобальной настройки.

Д) Объединение точек

В случае отсутствия замыкания цикла, алгоритм РТАМ занят воссозданием ключевых точек в предыдущих кадрах (этим обусловлено то, что одна точка пространства может соответствовать нескольким точкам), поэтому после замыкания цикла алгоритм удаляет все повторяющиеся точки.

Начиная с текущего кадра $T_{k,w}$, необходимо проецировать на него все точки, увиденные в замыкающем кадре цикла $T_{l,w}$, стирая всё, что указывает

на текущий кадр, находящиеся в пределах 20 пикселей от проецируемой точки. Затем необходимо повторить, только проецируем теперь точки из кадра $T_{l - \delta, w}$ и $T_{l + \delta, w}$ до тех пор, пока ни одна точка не удаляется. Как только все ненужные точки удаляются, необходимо ввести соответствия между точками везде, где точки были удалены. Используется поиск по достаточно большому радиусу, так как соединения между точками еще не оптимизированы[12].

Е) Исправление траектории

Финальный этап закрытия цикла заключается в исправлении положения камеры и её скорости и информировании рабочей системы о закрытие цикла. Изначально требуется вычисление относительного изменения текущего положения камеры $T_{c, w}$ к неисправленному положения текущего кадра $T_{k, w}$

$$\Delta T_{c, k} = T_{c, w} * T_{k, w}^{-1}$$

После следует этап масштабирования трансформации $\Delta T_{c, k}$ по коэффициенту масштаба $s_{k, l}$:

$$\Delta T_{c, k} = \begin{bmatrix} R_{c, k} & t_{c, k} \\ 0 & 1 \end{bmatrix} \rightarrow \Delta T_{c, k}^{cor} \begin{bmatrix} R_{c, k} & \frac{1}{s_{k, l}} t_{c, k} \\ 0 & 1 \end{bmatrix}$$

Следующий этап – для получения положения камеры необходимо применение $cor \Delta T_{c, k}$ к исправленному текущему положению кадра:

$$T_{c, k}^{cor} = \Delta T_{c, k}^{cor} * T_{k, w}^{cor}$$

Наконец нужно исправить скорость затухания движения. Делим линейную скорость на $s_{k, l}$.

Общий принцип работы программы и его логика представлена в блок-схеме на рисунке 5. Алгоритм одновременно работает в трех направлениях: отслеживание положений камеры и её координаты; замыкание циклов, где производится поиск кандидатов на замыкания цикла; локализация карты и её отрисовка, здесь обрабатываются новые ключевые кадры, и выполняется блок Bundle Adjustment. Если есть точки ORB для которых на предыдущих кадрах

не было соответствий, то они ищутся тут и на всех соединённых кадрах графе взаимовидимости[23].

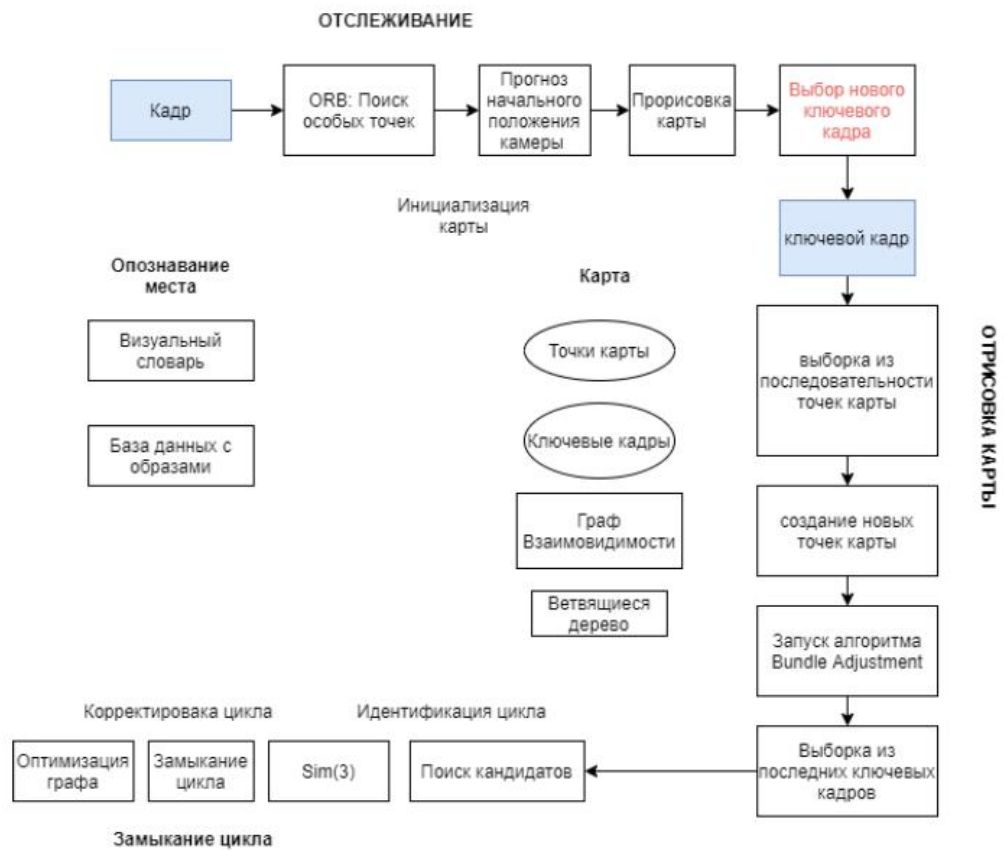


Рисунок 5 Схема алгоритма

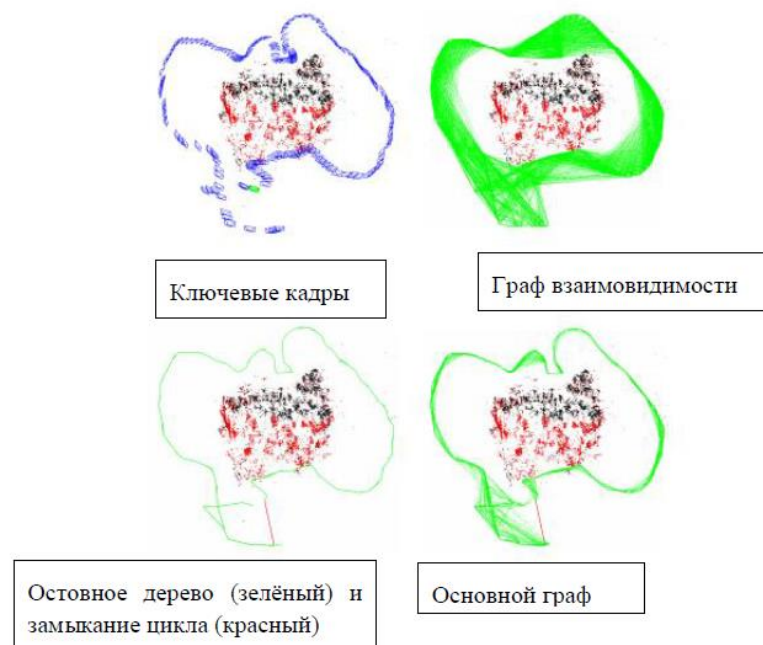


Рисунок 6 Виды графов в алгоритме

Важной частью алгоритма является блок Bundle Adjustment. Здесь происходит прогнозируемое перемещение камеры на некоторое расстояние. Происходит этот прогноз по формуле:

$$\min \sum_{i=1}^n \sum_{j=1}^m v_{i,j} d(Q(a_j, b_i), x_{i,j})^2$$

Здесь предполагается, что некоторое количество пространственных точек находятся на m изображениях, а $x_{i,j}$ это проекция i точки на j фотографии. $v_{i,j}$ обозначает бинарную функцию, принимающая два значения: 1 или 0, если точка i заметна на j фотографии, то значение функции равно 1, в противном случае - 0. Предположим, ещё что каждая камера j параметризована вектором a_j , а каждая 3D точка i параметризована вектором b_i . Этот алгоритм локализован в рамках минимальной ошибки, минимизирует ошибку положения камеры и проекции точек методом наименьших квадратов. В формуле $Q(a_j, b_i)$ это проекция точки i на кадр j , а $d(x, y)$ обозначает Евклидово расстояние между двумя векторами. $Q(a_j, b_i)$ в данном случае нелинейными методами, такими как метод Ньютона, или улучшенной его версией – методом Левенберга –Марквардта решается данное нелинейная операция.

Расчёт расстояния передвижения робота относительно предыдущего ключевого кадра зависит от постоянства относительно предыдущего кадра скорости передвижения. С помощью модели RANSAC вычисляется их взаимное расположение, а 3D точки, найденные на текущем кадре, сопоставляются с теми, что были найдены ранее.

Заключение к 3 главе

В данной главе был проведен обзор и анализ конкретно одной реализации алгоритма локальной навигации и картографирования monoSLAM – ORBSLAM. Был проведен обзор принципа работы дескриптора особых точек ORB, который является отличным дескриптором, так как не теряет точности при ротации изображения (например, актуально для реализации навигационной системы на базе монокулярной камеры квадрокоптера), также описан принцип работы самого ORBSLAM. Данный алгоритм выбран для разработки макета тестового программного обеспечения, так как он удовлетворяет все условия: совместим с камерой мобильного робота и реализуем на его бортовом компьютере.

4 Разработка макета программного обеспечения на основе алгоритма ORBSLAM и проведение комплексных экспериментальных исследований по оценке работоспособности созданного макета программного обеспечения

4.1 Мобильный автономный робот NVIDIA JetBot

Испытания макета разработанной системы навигации на базе монокулярного ORBSLAM будут проводиться на роботе NVIDIA JetBot. Первоначально планируемое место проведения испытаний - Лаборатория “Мобильная робототехника” аудитория Г210, однако в связи с техническими причинами пришлось ограничиться реализацией в среде симулятора Gazebo.



Рисунок 8 Физическая модель робота NVIDIA JetBot

NVIDIA JetBot – это автономный мобильный двухколесный робот с открытым исходным кодом, основанный на базе небольшого, но довольно мощного компьютера NVIDIA Jetson Nano, технические характеристики которого представлены на рисунке 9.

| | |
|---------------------|--|
| GPU | 128-core Maxwell |
| CPU | Quad-core ARM A57 @ 1.43 GHz |
| Memory | 4 GB 64-bit LPDDR4 25.6 GB/s |
| Storage | microSD (not included) |
| Video Encode | 4K @ 30 4x 1080p @ 30 9x 720p @ 30 (H.264/H.265) |
| Video Decode | 4K @ 60 2x 4K @ 30 8x 1080p @ 30 18x 720p @ 30 (H.264/H.265) |
| Camera | 1x MIPI CSI-2 DPHY lanes |
| Connectivity | Gigabit Ethernet, M.2 Key E |
| Display | HDMI 2.0 and eDP 1.4 |
| USB | 4x USB 3.0, USB 2.0 Micro-B |
| Others | GPIO, I ² C, I ² S, SPI, UART |
| Mechanical | 100 mm x 80 mm x 29 mm |

Рисунок 9 Технические характеристики бортового компьютера JetBot’a

Так как принято решение проводить экспериментальные исследования в среде симуляторе Gazebo то, следовательно, необходимо указать кинематические уравнения для описания движения и перемещения математической модели мобильного робота[15]:

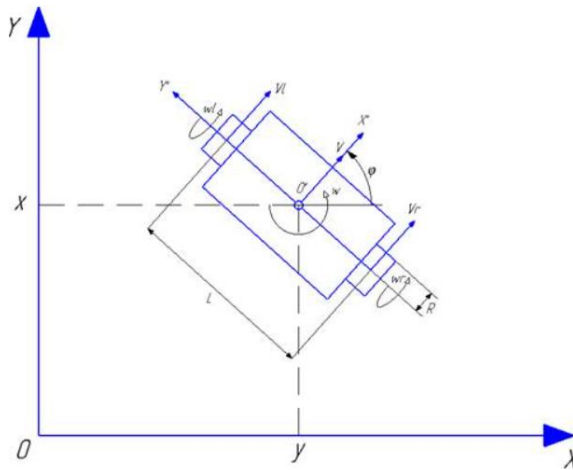


Рисунок 9 Кинематическая модель двухколесного робота

$$\begin{cases} \dot{x} = v * \cos \varphi \\ \dot{y} = v * \sin \varphi \\ \dot{\varphi} = \omega \\ v = \frac{R}{2}(\omega_r + \omega_l) \\ \omega = \frac{R}{2}(\omega_r - \omega_l) \end{cases}$$

$$\begin{cases} \omega_l = \frac{1}{R}\left(v - \frac{\omega l}{2}\right) \\ \omega_r = \frac{1}{R}\left(v + \frac{\omega l}{2}\right) \end{cases}$$

$$\begin{cases} \Delta S = \frac{R(\Delta\alpha_R + \Delta\alpha_L)}{2} \\ \Delta\varphi = \frac{R(\Delta\alpha_R - \Delta\alpha_L)}{2l} \end{cases}$$

где x и y – координаты модели на плоскости, v и ω – линейная и угловая скорость модели, φ - угол, характеризующий направление движения объекта относительно оси ординат, l – половина расстояния между колесами, R – радиус колеса, $\Delta\alpha_R$, $\Delta\alpha_L$ - угол поворота правого и левого колеса робота, соответственно, ΔS – перемещение робота и $\Delta\varphi$ – угол поворота робота относительно точки O .

4.2 Среда симулятора Gazebo

Средой, в которой проводились все экспериментальные исследования, является Gazebo - робототехнический симулятор, поддерживающий среду Robot Operation System. Симулятор состоит из графической части и части по имитированию взаимодействия твердых объектов, позволяя моделировать динамику и кинематику механизмов роботов (включая моменты взаимодействия с телами внешней среды) и формировать физически правдоподобные показания виртуальных датчиков.

По умолчанию, модель робота NVIDIA JetBot не предустановлена в среде симулятора, но это можно сделать, установив пакет `jetbot_ros`.

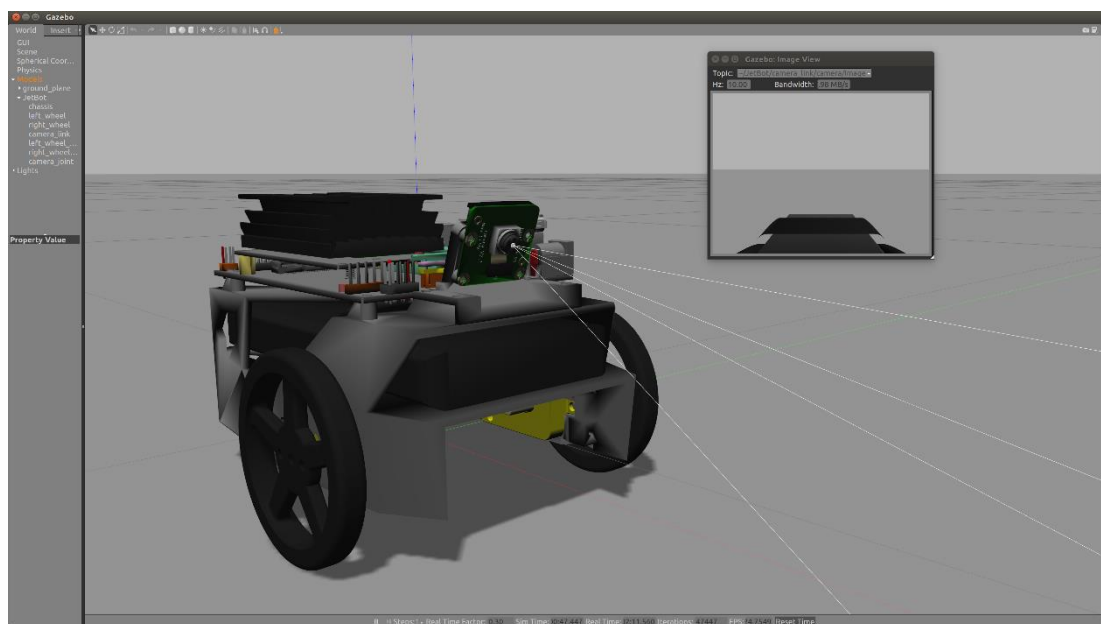


Рисунок 10 Модель робота в симуляторе Gazebo

4.3 Камера мобильного робота NVIDIA JetBot

На борту робота предустановлено 2 камеры: камера Intel RealSense d435 (рисунок 11а) и монокулярная камера типа «рыбий глаз» (fisheye) (рисунок 11б).



Рисунок 11а и 11б Камеры, установленные на борту JetBot`а

Алгоритм ORBSLAM можно было бы реализовать на базе камеры RealSense, программно отключив модуль камеры глубины, но в таком случае данная реализация экономически не выгодна, так как монокулярная камера значительно дешевле. При запуске данной камеры, получаются следующие изображения (рисунок 12а и 12б):



Рисунок 12а и 12б Изображения с монокулярной камеры робота

Объектив типа «рыбий глаз», благодаря большому углу обзора, точность визуальной одометрии. Угол обзора данной камеры 160 градусов. Но также присутствуют и минусы использования данной камеры. Как видно из данных изображений (рисунок 12а и 12б), что линза «рыбий глаз» искажает полученное изображение, а также неточно передается цветовая гамма. Но обе эти проблемы разрешимы. Искажение объективом нивелируется посредством грамотной калибровки, а неточная цветокоррекция не столь важна, так как алгоритм ORBSLAM полученное изображение конвертирует в черно-белый формат. Для камер с широкоугольными объективами или типа «рыбий глаз» для реализации алгоритма ORBSLAM калибровка применяется по модели KannalaBrandt8. Параметры камеры, которые необходимо указать, это фокусное расстояние (f_x, f_y) и центральная точка в пикселях (c_x, c_y) и четыре коэффициента для модели эквидистантных искажений (k_1, k_2, k_3, k_4). Калибровочный файл можно найти в приложении в программе calibration.

4.4 Создание макета программного обеспечения

Для получения более конкретного результата, а также упрощения создания макета ПО построим карту окружающей среды в среде симулятора Gazebo. Модель робота должна двигаться по замкнутой циклической траектории, для визуального сходства исходного изображения и построенной карты местности. Предполагаемая траектория движения робота продемонстрирована на рисунке 13.

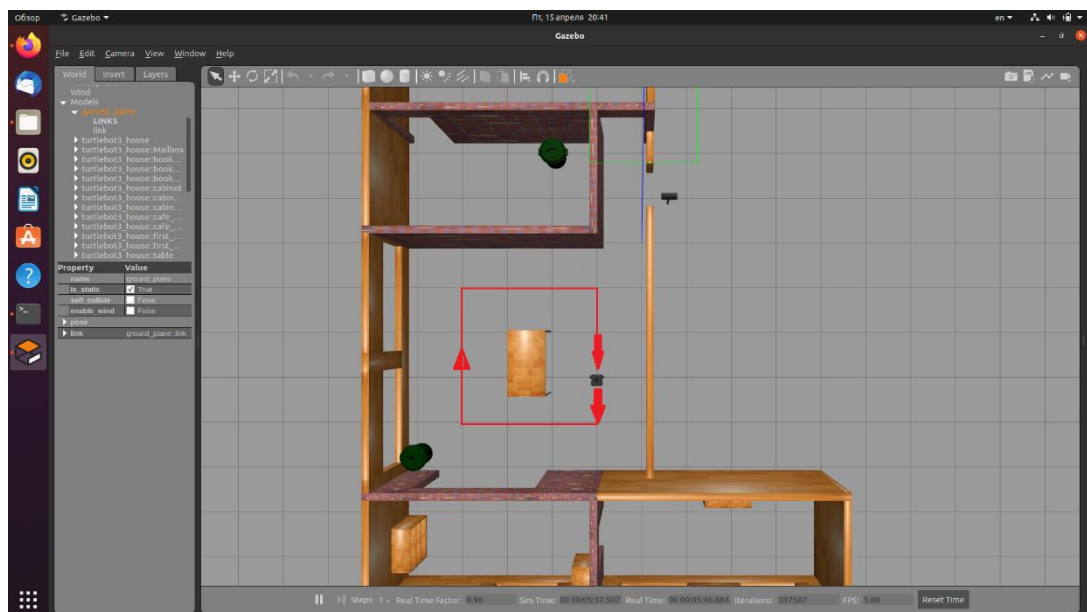


Рисунок 13 Замкнутая траектория движения робота

Запустим камеру робота в среде симулятора вызовом команды чтобы убедиться в работоспособности данного модуля (рисунок 14).

```
roslaunch image_view image_view image:=camera/rgb/image_raw,
```

В ROS есть пакет реализации алгоритма ORBSLAM - orb_slam2_ros. Но, как упоминалось ранее, крайне важна калибровка камеры, так как алгоритм подстраивается под особенности каждой камеры.

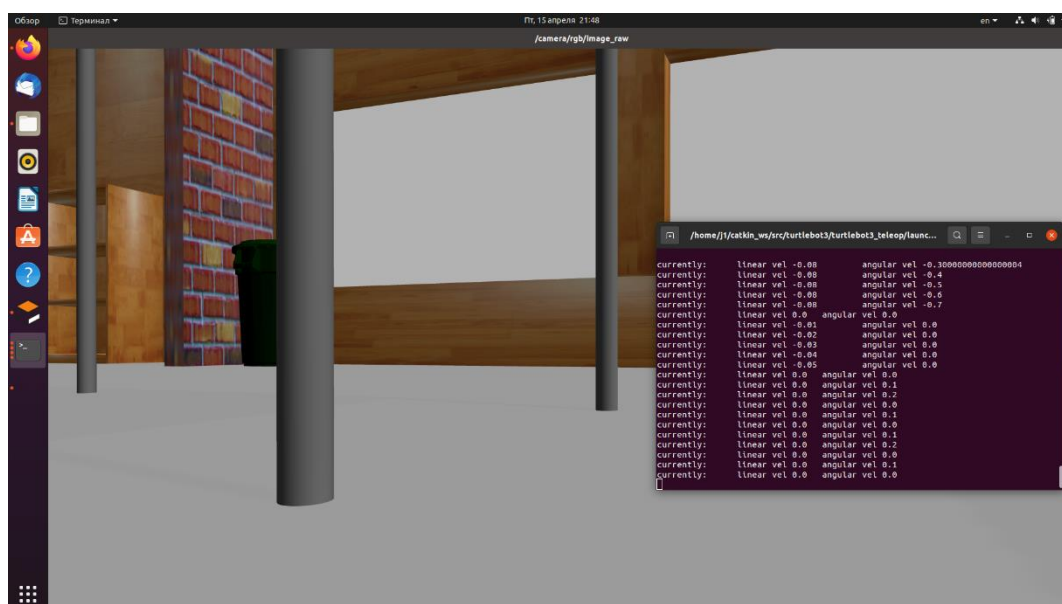


Рисунок 14 Изображение с RGB камеры робота

4.5 Проведение экспериментальных исследований

Чтобы задать движение робота, запустим файл `jetbot_teleop_key`, вызовом команды в терминале: `roslaunch jetbot_ros jetbot_teleop_key.launch`. После того, как собрали все пакеты внутри рабочего пространства `catkin_ws`, подключим алгоритм ORBSLAM, для этого в терминале указываем команду: `roslaunch orb_slam2_ros orb_slam2_r200_mono.launch`. В окне терминала выводится следующее сообщение: `Map point vector is empty!` (рисунок 15) Это значит, что недостаточно точек для калибровки камеры, для этого в Gazebo открываем Building Editor и добавляем текстур для упрощения калибровки камеры.

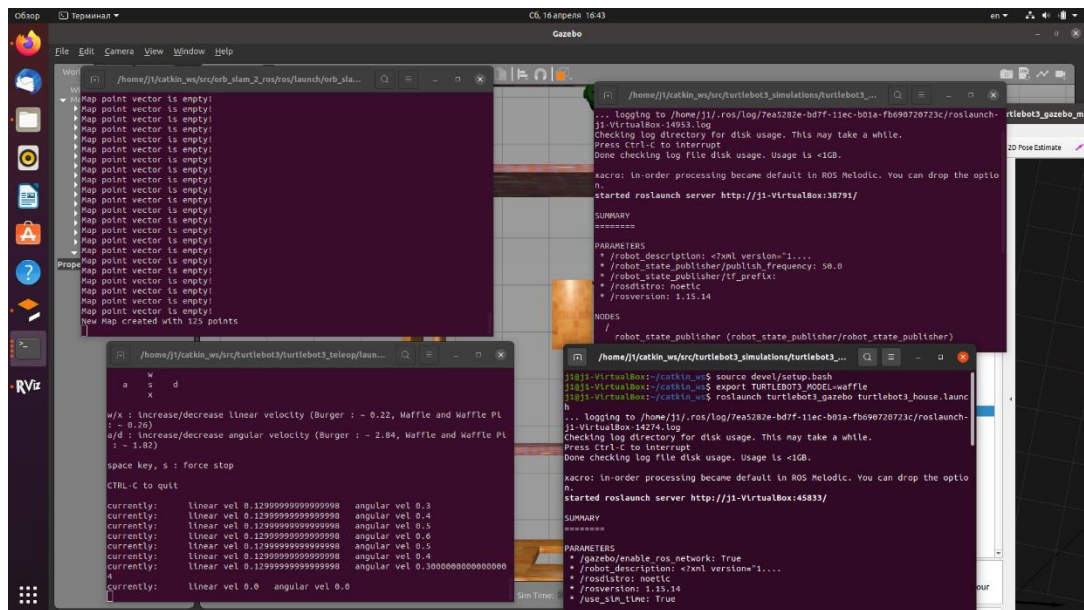


Рисунок 15 Окна запущенных программ

Запустив программу ORBSLAM в окне терминала, выведется 2 окна, одно - с найденными особыми точками на сцене (рисунок 16а), а в другом окне отображается рисуемая карта 3D пространства (рисунок 16б) с положением камеры.

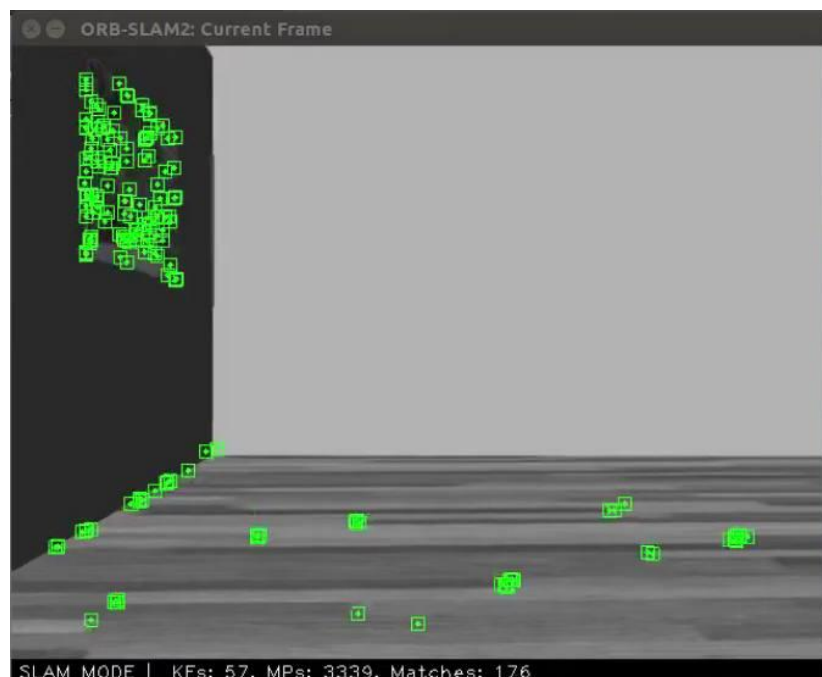


Рисунок 16а Поиск дескриптором ORB особых точек на конвертированном изображении

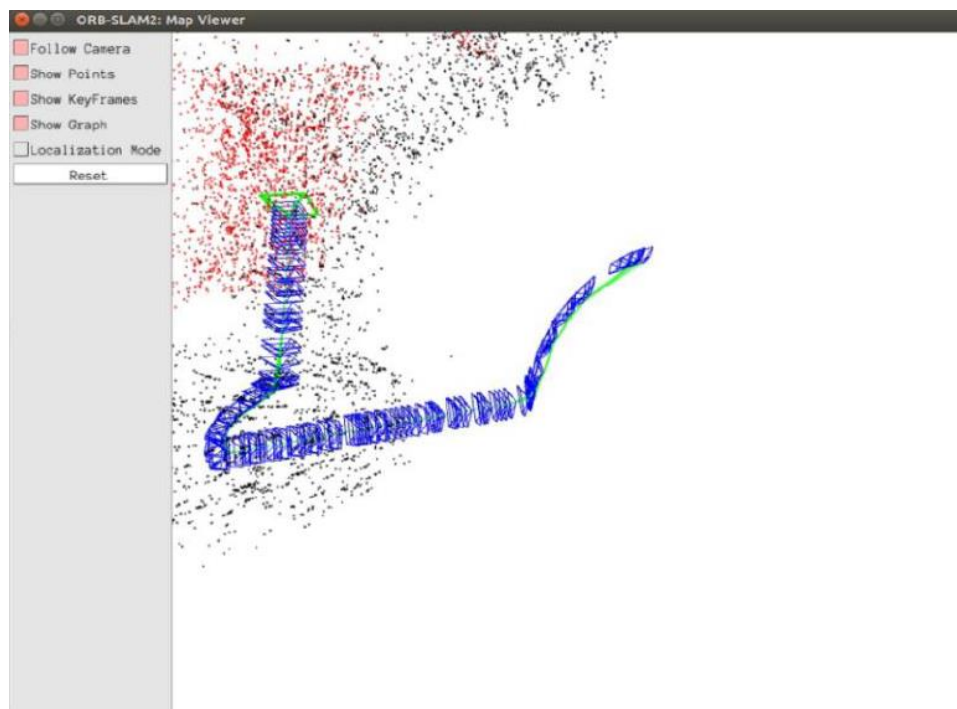


Рисунок 166 Отображаемая карта, выводимая в окне программы ORBSLAM

По прошествии какого-то времени начинает вырисовываться карта. Однако из-за накопленной ошибки она не совсем правильная. Это происходит из-за того, что в системе, в которой была бы инвариантность к масштабу, не оставалось бы неопределённости в правильности выставления 3D точек на карту, однако в монокулярной системе эта определённость остаётся. Масштаб выступает как дополнительная степень свободы, с которой приходится мириться и пытаться определить геометрию сцены исходя из последовательности кадров с одинаковыми особыми точками, что приводит к ошибкам масштабирования в некоторых сегментах карты. Поэтому так важно, чтобы применялся алгоритм оптимизации карты после завершённого цикла.

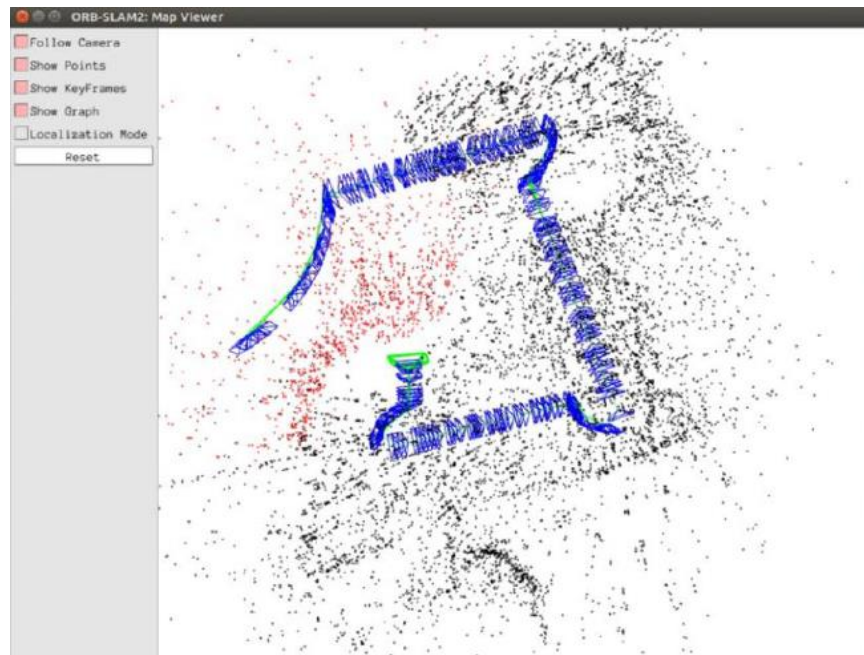


Рисунок 17 Процесс отрисовки карты

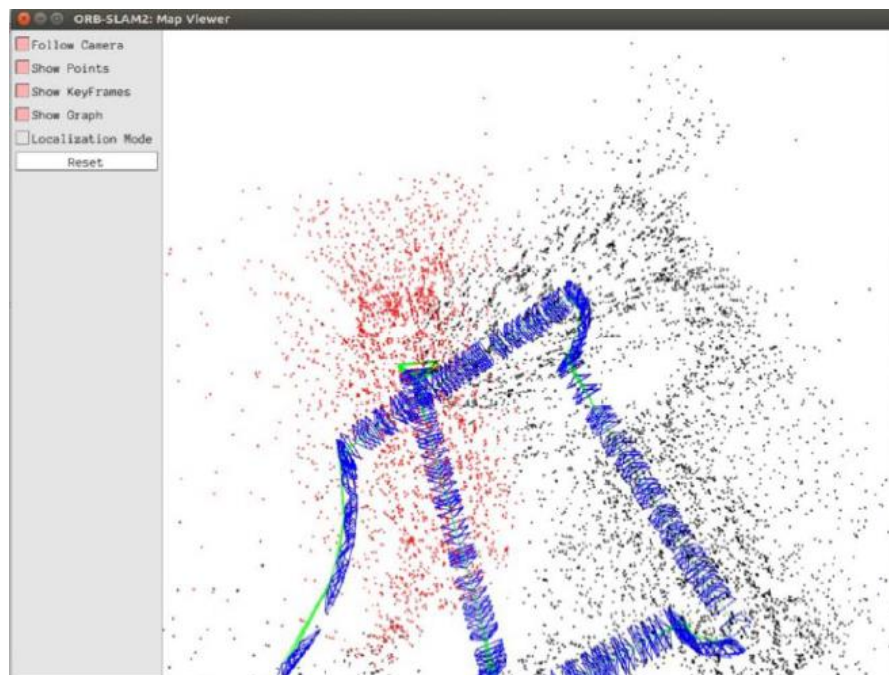


Рисунок 18 Накопившаяся ошибка в процессе картографирования

Как видно на рисунке 18 карта построилась неправильно, однако на следующем изображении видно, что происходит корректировка прогнозируемого положения точек карты. Это происходит из-за трансформации подобия, получив информацию о том, что цикл закрылся,

начинается подсчёт накопленной ошибки и она распределяется по все фреймам графа.

По ходу движения с каждым новым прохождением по той же траектории карта становится всё более корректной.

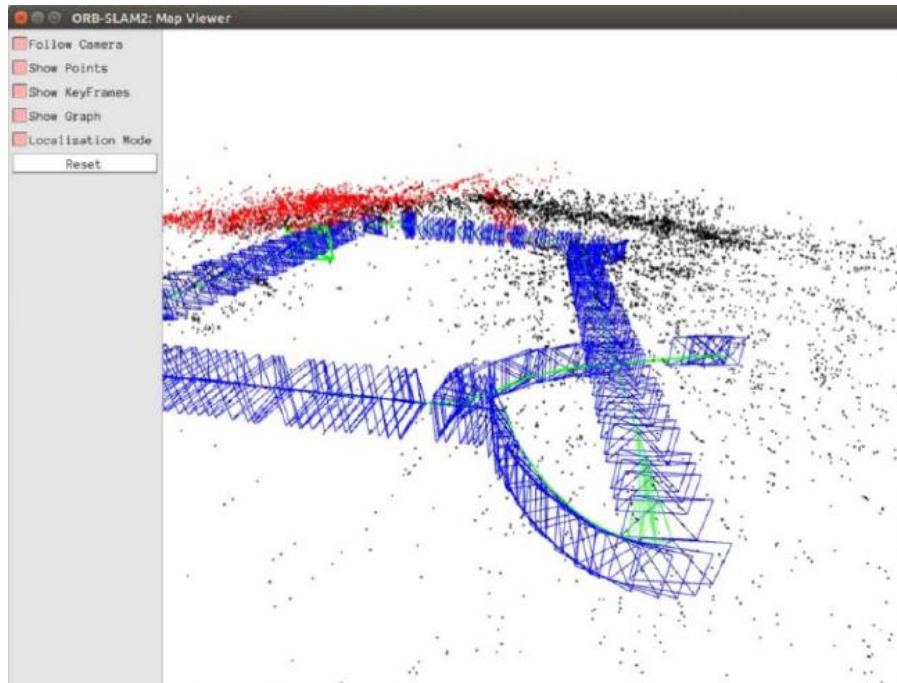


Рисунок 19 Корректировка карты путем повторных прохождений

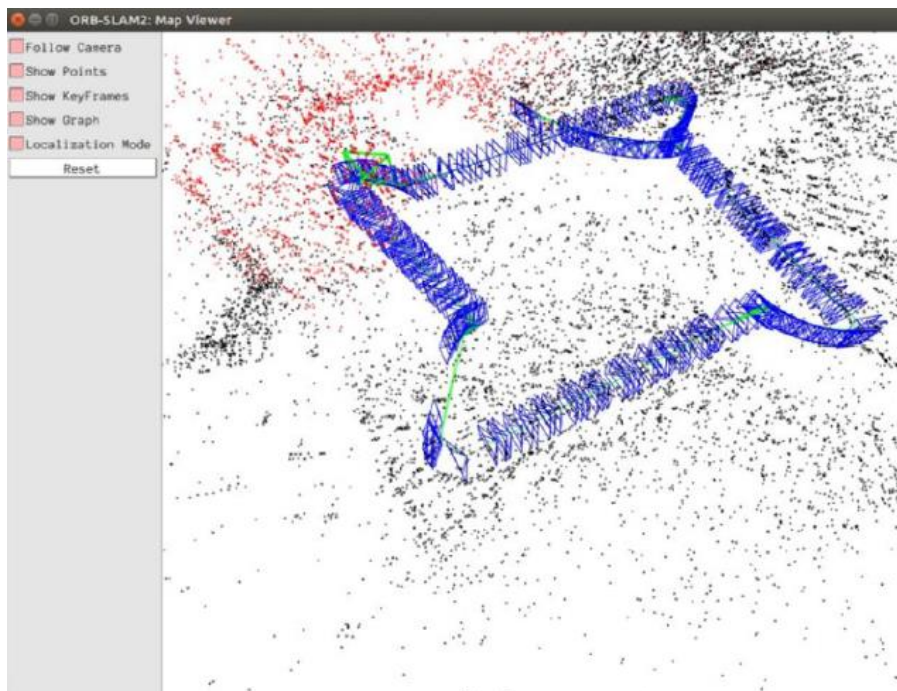


Рисунок 20 Построенная карта среды симулятора

По результатам экспериментальных исследований был продемонстрирован принцип картографирования, реализованный макетом разработанного ПО, простой карты по замкнутой траектории в среде симулятора Gazebo.

4.6 Критерии качества карты местности, построенной с помощью алгоритма

В первую очередь производится визуальное сравнение полученного изображения карты с исходным изображением карты местности[3]. Это позволяет тестировщику качественно оценить работу алгоритма и выявить самые заметные проблемы в работе SLAM. Из таких проблем можно выделить, например: пропуски каких-либо объектов на карте, определение одной и той же стены как двух при обходе с разных сторон, неправильное сшивание разных частей карты. Присутствие таких ошибок не позволяет количественно оценить точность карты, поэтому, прежде чем переходить к количественной оценке, необходимо произвести настройку алгоритма SLAM, частоты снятия данных с сенсоров, при необходимости, скорости движения робота, и т. д.

После удовлетворенности результатами качественной оценки (т. е. отсутствии вышеперечисленных ошибок), рассчитывается количественная точность работы SLAM с помощью среднеквадратичной ошибки траектории. Для этого необходимо произвести запуск программы несколько раз, и по записанным данным о положении робота из узлов вычислить среднеквадратичную ошибку. Понадобятся лишь узлы одометрии и времени для того, чтобы рассчитать среднеквадратическую ошибку траектории, которая равна:

$$RMSE = \sqrt{MSE(X)} = \sqrt{\frac{1}{n} \sum_{i=1}^n (X_i - \hat{X}_i)^2}$$

где X_i – реальное значение положения/ориентации робота в пространстве в момент времени i , X_i – расчетное значение положения/ориентации робота, n – количество измерений.

Альтернативной системой оценки точности построенной карты местности является формула расчета среднего и максимального отклонения координат углов на построенной карте от координат соответствующих углов на исходной карте[20].

Формула для расчета отклонения координат построенной карты от координат реальной карты:

$$r_i = \sqrt{(x_{1i} - x_{2i})^2 + (y_{1i} - y_{2i})^2}$$

Где x_{1i} и y_{1i} - координаты i -го угла на реальной карте, а x_{2i} и y_{2i} - соответствующие координаты на построенной карте. Чем меньше значение r_i , тем достовернее получилась построенная карта местности.

Заключение к 4 главе

В четвертой главе был разработан макет программного обеспечения навигационной системы на базе алгоритма ORBSLAM для мобильного робота NVIDIA JetBot с учетом особенности бортовой RGB камеры робота. В ходе экспериментальных исследований был продемонстрирован общий принцип работы алгоритма внутри довольно тривиального пространства. Важной особенностью алгоритма локальной навигации является тот факт, что в нем используется монокулярный SLAM, что чревато комплексным способом распознавания образов и 3D точек для определения геометрии окружения, однако в ходе экспериментальных исследований, несмотря на однообразие моделируемой среды, (усложнение для ORB в поиске дескрипторов) были выявлены алгоритмом ORB особые точки.

Построенная карта имеет визуальное сходство с первоначальной траекторией движения, очертание квадрата явно проглядывается, но вместе с этим присутствуют и недочеты, такие как повторное определение ранее пройденного участка.

Определение количественного значения точности остается задачей дальнейшего изучения, так как разработанный макет программного обеспечения работает, но требует значительной доработки.

5 Организационно-экономическая часть

5.1 Введение

В современном мире любой продукт будет иметь успех на рынке при соблюдении баланса цены и спроса. Разрабатываемое ПО является высокотехнологичным продуктом. Как и у любого продукта, у него есть себестоимость и цена продажи. Для успешного позиционирования на рынке необходимо рассчитать цену продажи таким образом, чтобы она соответствовала ожиданиям потребителя.

Согласно заданию, на выпускную квалификационную работу, целями выполнения данного раздела являются:

- Рынок сбыта
- Конкурентоспособность
- Маркетинг
- Организация и планирование работ по теме.
- Определение договорной цены.
- Оценка экономической целесообразности проведения работ по теме.

5.2 Разработка основных разделов бизнес-плана проекта

5.2.1 Описание продукта

Конечный продукт представляет собой ПО для робототехнического комплекса, состоящего из робота NVIDIA JetBot, оснащённого системой технического зрения. Эта программа предназначена для картографирования зоны работы робота. Цели продукта могут варьироваться. Спектр возможных

задач, решаемых роботом могут заключаться в ознакомлении студентами вуза с алгоритмами локальной навигации, так и применение продукта в масштабах коммерческих организаций, в таком случае заказчиком будут эти организации.

5.2.2 Анализ рынка сбыта

Представленный продукт (алгоритм) предназначен для внутреннего использования кафедрой проблем управления Института искусственного интеллекта федерального государственного бюджетного образовательного учреждения высшего образования «Российский технологический университет» МИРЭА.

5.2.3 Конкурентоспособность

Конкурентоспособность — это такая способность, которую имеет определённый объект или субъект, отвечающий запросам заинтересованных лиц по сравнению с другими такими же субъектами и/или объектами. Объекты могут быть товарами, предприятиями, отраслями, регионами (страны, области, районы). Субъекты могут быть потребителями, производителями, государством, инвесторами. Конкурентоспособность может быть определена, только путём сравнения объектов или субъектов с другими между собой.

Данный алгоритм разрабатывался под требования кафедры проблем управления института искусственного интеллекта ФГБОУ ВО «Российский технологический университет» МИРЭА. Таким образом, данный алгоритм учитывает специфику только тех требований, которые были предъявлены Заказчиком со стороны кафедры проблем управления.

На данный момент существует множество алгоритмов навигации и картографирования местности, однако у этого алгоритма есть большие преимущества перед ними. В первую очередь это быстрая производительность и небольшие затраты вычислительной мощности. Поэтому этот продукт обладает высокой конкурентоспособностью.

5.2.4 План маркетинга

Главным способом привлечения потребителей является реклама. Распространение рекламы можно реализовать в рамках ФГБОУ ВО «Российский технологический университет» МИРЭА следующими способами:

- Размещение информации о данном алгоритме на официальном сайте МИРЭА и сайте кафедры проблем управления;
- Доведение информации о системе до сотрудников посредством рассылки на электронную почту института.

5.3 Организация и планирование работ по теме

В составе работы задействовано 3 человека:

- руководитель (Кучерский Роман Владимирович, доцент, кафедра Проблем управления) – человек, отвечающий за грамотную постановку задачи, контролирующий отдельные этапы работы, он вносит необходимые корректировки и оценивает выполненную работу в целом;
- консультант (Голубов Владимир Васильевич, ассистент, кафедра Проблем управления) – человек, который консультирует в области технической части проекта: предлагает возможные пути решения задач, выбора инструментов разработки, способов разработки;
- разработчик (студент 4-го курса Санин Никита Максимович, КРБО-02-18) – человек, который реализует все поставленные задачи, а также проводит тестирование готового продукта и подготавливает проектную документацию.

Состав задействованных в работе участников представлен на схеме.



Рисунок 20 Состав задействованных в проекте участников

5.3.1 Организация работ

На разработку отводится 60 рабочих дней. Этапы разработки представлены в таблице 5.1

| | Название этапа | Исполнитель | Трудоемкость, чел/дни | Продолжительность работ, дни |
|-----|---|--------------|-----------------------|------------------------------|
| 1 | Разработка и утверждение технического задания | Руководитель | 2 | 2 |
| 2 | Технические предложения | Руководитель | 3 | 3 |
| | | Консультант | 2 | |
| 3 | Эскизный проект: | | | 13 |
| 3.1 | Анализ исходных данных и требований | Разработчик | 6 | |
| 3.2 | Постановка задачи | Консультант | 2 | |
| 3.3 | Разработка общего описания алгоритма функционирования | Руководитель | 2 | |
| | | Разработчик | 5 | |
| 4 | Технический проект: | | | 8 |
| 4.1 | Определение формы представления входных и выходных данных | Руководитель | 2 | |
| | | Разработчик | 3 | |
| 4.2 | Разработка структуры программы и логической структуры базы данных | Руководитель | 2 | |
| | | Консультант | 2 | |
| | | Разработчик | 5 | |
| 5 | Рабочий проект: | | | 34 |
| | Программирование и отладка | Разработчик | 18 | |

| | | | | |
|-------|--|--------------|----|----|
| 5.1 | программы | | | |
| 5.2 | Испытание программ ы | Разработчик | 3 | |
| 5.3 | Корректировка программы по результатам испытаний | Разработчик | 4 | |
| 5.4 | Подготовка технической документации и апрограммный продукт | Консультант | 3 | |
| | | Разработчик | 5 | |
| 5.5 | Сдача готового продукта и внедрение | Руководитель | 2 | |
| | | Консультант | 2 | |
| | | Разработчик | 4 | |
| Итого | | | 79 | 60 |

Общий принцип работы программы представлен на рисунке 21

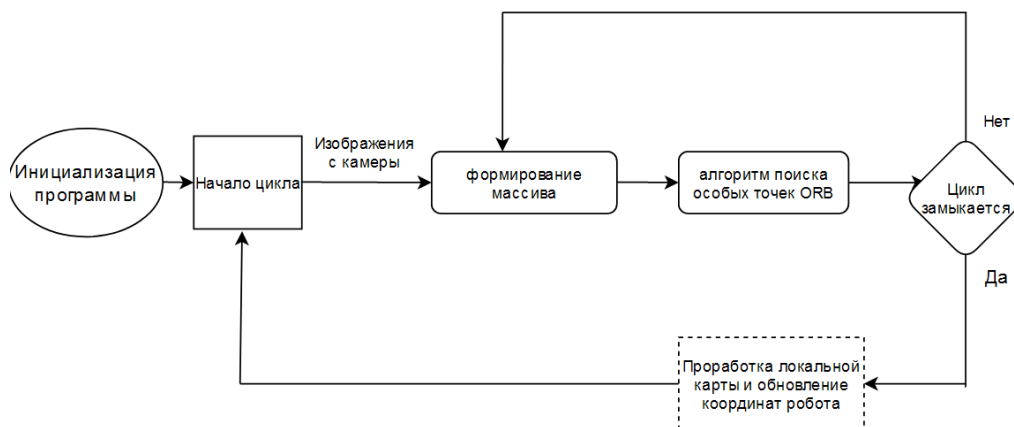


Рисунок 21 Календарный график выполнения работ

5.3.2 График проведения работ

График проведения работ представлен на рисунке 22 и составляет 60 дней.

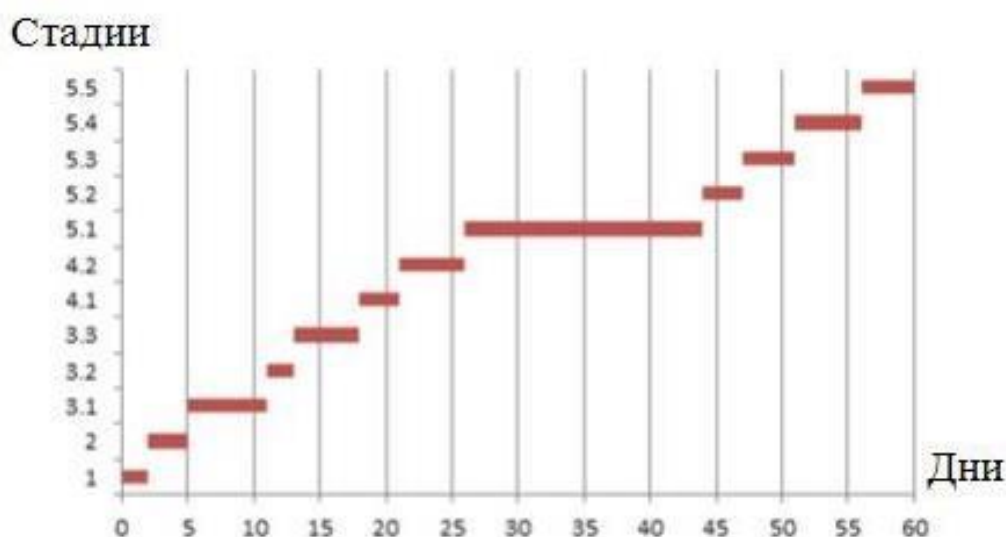


Рисунок 22 Календарный график выполнения работ

5.3 Определение договорной цены

1 статья «Материалы, покупные изделия и полуфабрикаты + ТЗР (15%) от Σ итога по материалам.

2 статья «Специальное оборудование» - как правило, затрат нет.

3 статья «Основная заработная плата».

4 статья «Дополнительная заработная плата» 20% от основной заработной платы.

5 статья «Страховые отчисления» - 30% от ФОТ.

6 статья «Командировочные расходы» - как правило, затрат нет.

7 статья «Контрагентские услуги» - как правило, затрат нет.

8 статья «Накладные расходы» - 250% от основной заработной платы.

9 статья «Прочие расходы» - затрат нет.

В выпускной квалификационной работе объем затрат на НИР и ОКР был проведен методом калькулирования.

1 статья «Материалы, покупные изделия и полуфабрикаты»

| № пп | Наименовани ематериалов | Единицы измерени я | Количество | Цена за единицу (руб) | Стоимость (руб) |
|-------------------------------------|----------------------------|--------------------------|------------|-----------------------------|--------------------|
| 3 | Бумага А4 | пачка | 3 | 250 | 750 |
| 4 | Картридж для принтера | шт | 1 | 2000 | 2000 |
| 5 | Ручка | шт | 10 | 25 | 250 |
| 6 | Карандаш | шт | 2 | 10 | 20 |
| 7 | 2950 | | | | |
| Итого материалов | | | | | 3020 |
| Транспортно-заготовительные расходы | | | | | 443 |
| Итого | | | | | 3463 |

Транспортно-заготовительные расходы рассчитываются как 15% от общей суммы денежных средств на материальные и покупные изделия и будут составлять 443 руб. Стоимость продукта, учитывая транспортно-заготовительные расходы, будет составлять 3463 руб.

2 статья «Специальное оборудование»

Расходов на специальное оборудование нет, так как все разработки выполняются на базе оборудования МИРЭА «Российский технологический университет».

3 статья «Основная заработная плата»

Расчёт основной заработной платы представлен в таблице 5.3. Оплата за день рассчитывается путём деления месячного оклада на 22 рабочих дня.

Таблица 5.3 – Расчёт основной заработной платы

| № пп | Исполнитель (должность) | Мес. окла д (руб) | Трудоём- кость (чел/дни) | Оплата за день (руб) | Оплата за этап (руб) |
|---------|----------------------------|----------------------------|--------------------------------|-------------------------|-------------------------|
| 1 | Руководитель | 40 000 | 3 | 1818 | 5454 |
| 2 | Руководитель | 40 000 | 3 | 1818 | 5454 |
| | Консультант | 30 000 | 2 | 1591 | 3182 |
| 3 | Руководитель | 40 000 | 2 | 1818 | 3636 |
| | Консультант | 30 000 | 2 | 1591 | 3182 |
| | Разработчик | 25 000 | 11 | 1318 | 14498 |
| 4 | Руководитель | 40 000 | 4 | 1818 | 7272 |
| | Консультант | 30 000 | 2 | 1591 | 3182 |
| | Разработчик | 29 000 | 7 | 1318 | 9226 |
| 5 | Руководитель | 40 000 | 2 | 1818 | 3636 |
| | Консультант | 30 000 | 6 | 1591 | 9546 |
| | Разработчик | 29 000 | 34 | 1318 | 44812 |
| Итого | | | | | 113080 |

4 статья «Дополнительная заработная плата»

Дополнительная заработная плата составляет 20% от суммы основной заработной платы.

$$\text{ДЗП} = 113\,080 \times 0,2 = 22\,616 \text{ руб.}$$

Дополнительная заработная плата научного и производственного персонала будет составлять по проекту 22 616 руб.

5 статья «Страховые отчисления»

Отчисления на социальные нужды составляют 30% (22% - взносы в Пенсионный фонд, 5,1% - фонд обязательного медицинского страхования (ОМС), 2,9% - взносы в фонд социального страхования) от фонда оплаты труда (ФОТ), состоящий из основной и дополнительной заработной платы.

$$\text{ФОТ} = \text{ОЗП} + \text{ДЗП} = 113\,080 + 22\,616 = 135\,696 \text{ руб.}$$

$$\text{СВ} = \text{ФОТ} \times 30\% = 135696 \times 0,30 = 40\,708 \text{ руб.}$$

6 статья «Командировочные расходы»

Расходы по данному разделу отсутствуют, так как в ходе проекта командировок не осуществлялось.

7 статья «Контрагентские услуги»

В процессе разработки данного проекта не были использованы услуги сторонних организаций.

8 статья «Накладные расходы»

Накладные расходы:

$$\text{НР} = \text{ОЗП} \times 250\% = 113\,080 \times 2,5 = 339\,240 \text{ руб.}$$

9 статья «Прочие расходы»

По статье «прочие расходы» затрат нет.

10 статья «Налог на добавленную стоимость (НДС)»

Разработка ведется на базе кафедры проблем управления института искусственного интеллекта РТУ МИРЭА «Российский технологический университет», поэтому НДС не взимается.

В таблице 5.4. представлен расчёт полной себестоимости проекта.

| № пп | Номенклатура статей расходов | Затраты (руб) |
|---------|--|---------------|
| 1 | Материалы, покупные изделия и полуфабрикаты (за вычетом отходов) | 3463 |
| 2 | Специальное оборудование для научных(экспериментальных) работ | - |
| 3 | Основная заработная плата научного и производственного персонала | 113080 |
| 4 | Дополнительная заработная плата научного и производственного персонала | 22616 |
| 5 | Страховые взносы в социальные фонды | 40708 |
| 6 | Расходы на научные и производственные командировки | - |
| 7 | Оплата работ, выполненных сторонними организациями и предприятиями | - |
| 8 | Прочие прямые расходы | - |
| 9 | Накладные расходы | 339240 |
| 10 | НДС | - |
| Итого | | 519107 |

Предположительно, продукт в дальнейшем будет реализован, тогда необходимо рассчитать договорную цену.

Цена договорная = себестоимость + прибыль + НДС
 Норма прибыли составляет 30% от стоимости разработки. Прибыль будет равна: $P = 519107 * 30\% = 155\,732$ руб.

$$\text{НДС} = (C + \Pi) \times 20\% = (519\,107 + 155\,732) \times 20 : 100 = 134\,967 \text{ руб.}$$

Таким образом, договорная цена составит:

$$\text{ДЦ} = C + \Pi + \text{НДС} = 519\,107 + 155\,732 + 134\,967 = 809\,806 \text{ руб.}$$

5.4 Оценка экономической целесообразности проведения работ

Экономическая целесообразность разработки алгоритмов картографирования местности и локальной навигации мобильного робота заключается в следующем:

- Основная экономическая характеристика этого проекта — это норма прибыли в 113 080 рублей. По итогам расчёта полная себестоимость проекта составила 519 107 руб. Договорная цена на разработку с учётом 30% прибыли составила 809 806 руб;
- Разработанный алгоритм является уникальным, полностью адаптированным к условиям и нуждам Заказчика, учитывающим специфику и перспективы развития предприятия Заказчика;
- Данный алгоритм достаточно дешёвый и имеет понятный интерфейс, который будет интуитивно понятен пользователю данного алгоритма;
- Разработанный алгоритм позволит повысить качество знаний, получаемых студентами кафедры проблем управления при изучении манипуляционных и промышленных роботов;
- Разработка данного алгоритма является экономически целесообразной.

Заключение к 5 главе

В пятой главе выпускной квалификационной работы был разработан бизнес-план проекта на разработку, создание и проверку на работоспособность алгоритма картографирования местности на базе робота NVIDIA JetBot, оснащённого системой технического зрения; был произведен расчёт трудозатрат работников и составлена смета затрат на выполнение проекта; далее было проведено технико-экономическое обоснование целесообразности проекта. Договорная цена на разработку с учетом 30% прибыли составила 809 806 руб. Использование данного алгоритма не только практично, но и экономически целесообразно для кафедры проблем управления института искусственного интеллекта ФГБОУ «Российский технологический университет» МИРЭА.

На сегодняшний день имеется масса алгоритмов картографирования местности на базе автономного робота, однако система технического зрения, доминирующая на сегодняшнем рынке, как правило, состоит из пары стереокамер, что конечно эффективно, но экономически невыгодно и уже не актуально. Продукт, описанный здесь обладает высокой производительностью и не требует высокой вычислительной мощности. По итогам расчета полная себестоимость проекта составила 519 107 руб.

Список литературы

- 1) G. Klein and D. Murray, "Parallel tracking and mapping for small AR workspaces," in IEEE and ACM International Symposium on Mixed and Augmented Reality (ISMAR), Nara, Japan, November 2007, pp. 225–23
- 2) GitHub [Электронный ресурс] ORB_SLAM2
URL: https://github.com/raulmur/ORB_SLAM (Дата обращения 15.05.2022)
- 3) CYBERLENINKA [Электронный ресурс] Алгоритмы локальной навигации и картографии для бортовой системы управления автономного мобильного робота
URL: <https://cyberleninka.ru/article/n/algoritmy-lokalnoy-navigatsii-i-kartografii-dlya-bortovoy-sistemy-upravleniya-avtonomnogo-mobilnogo-robota>
(Дата обращения 15.05.2022)
- 4) D. Galvez-L ´opez and J. D. Tard ´os, "Bags of binary words for fast ´ place recognition in image sequences," IEEE Transactions on Robotics, vol. 28, no. 5, pp. 1188–1197, 2012
- 5) Arxiv.org [Электронный ресурс] ORB-SLAM3: An Accurate Open-Source Library for Visual, Visual-Inertial and Multi-Map SLAM
URL: <https://arxiv.org/pdf/2007.11898.pdf> (Дата обращения 15.05.2022)
- 6) Mur-Artal, R. and Tard ´os, J. D. (2017). ORB-SLAM2: An open-source SLAM system for monocular, stereo and RGB-D cameras. IEEE Transactions on Robotics, 33(5):1255–1262.
- 7) CSDN [Электронный ресурс] ORB-SLAM2详解（二）代码逻辑
URL: <https://blog.csdn.net/u010128736/article/details/53169832>
(Дата обращения 15.05.2022)
- 8) HacAdda [Электронный ресурс] Using ORB-SLAM + ROS to map the world.
URL: <https://www.hackadda.com/post/2020/5/17/using-orb-slam-ros-to-map-the-world/> (Дата обращения 15.05.2022)
- 9) Habr [Электронный ресурс] Structure from motion
URL: <https://habr.com/ru/post/301522/> (Дата обращения 15.05.2022)
- 10) E. Rublee, V. Rabaud, K. Konolige, and G. Bradski, "ORB: an efficient alternative to SIFT or SURF," in IEEE International Conference on Computer Vision (ICCV), Barcelona, Spain, November 2011, pp. 2564–2571
- 11) D. Galvez-L ´opez and J. D. Tard ´os, "Bags of binary words for fast ´ place recognition in image sequences," IEEE Transactions on Robotics, vol. 28, no. 5, pp. 1188–1197, 2012
- 12) Fischler, M. A., & Bolles, R. C. (1987). Random Sample Consensus: A Paradigm for Model Fitting with Applications to Image Analysis and Automated

Cartography. Readings in Computer Vision, 726 – 740.doi:10.1016/b978-0-08-051581-6.50070-2

13) Habr [Электронный ресурс] Vision-based SLAM: монокулярный SLAM
URL: <https://habr.com/ru/company/singularis/blog/277537/> (Дата обращения 15.05.2022)

14) J. Engel, T. Schops, and D. Cremers, “LSD-SLAM: Large-scale direct monocular SLAM,” in European Conference on Computer Vision (ECCV), Zurich, Switzerland, September 2014, pp. 834–849.

15) Научно-издательский центр «Аспект» [Электронный ресурс] Разработка математической модели кинематики и динамики колесного дифференциального робота

URL: <https://na-journal.ru/1-2021-tehnika/2991-razrabotka-matematicheskoi-modeli-kinematiki-i-dinamiki-kolesnogo-differencialnogo-robota>

(Дата обращения 15.05.2022)

16) Waveshare [Электронный ресурс] JetBot AI

URL: <https://www.waveshare.com/catalog/product/view/id/3755>

(Дата обращения 15.05.2022)

17) hackster.io [Электронный ресурс] Controlling the jetbot robot from ROS

URL: <https://www.hackster.io/belochka/controlling-the-jetbot-robot-from-ros-014620> (Дата обращения 15.05.2022)

18) wikipedia.org [Электронный ресурс] Инерциальная навигация

URL: https://ru.wikipedia.org/wiki/%D0%98%D0%BD%D0%B5%D1%80%D1%86%D0%B8%D0%B0%D0%BB%D1%8C%D0%BD%D0%B0%D1%8F_%D0%BD%D0%B0%D0%B2%D0%B8%D0%B3%D0%B0%D1%86%D0%B8%D1%8F (Дата обращения 02.05.2022)

19) wikipedia.org [Электронный ресурс] Фильтр Калмана

URL: https://ru.wikipedia.org/wiki/%D0%A4%D0%B8%D0%BB%D1%8C%D1%82%D1%80_%D0%9A%D0%B0%D0%BB%D0%BC%D0%B0%D0%BD%D0%B0 (Дата обращения 05.05.2022)

20) Р.В. Кучерский, С.В. Манько Алгоритмы локальной навигации и картографии для бортовой системы управления автономного мобильного робота // 2012 с.10 -18

21) GitHub [Электронный ресурс] ORB_SLAM2-Python-Jetbot

URL: https://github.com/jerrywiston/ORB_SLAM2-Python-Jetbot

(Дата обращения 05.05.2022)

22) Habr [Электронный ресурс] Фильтр Калмана

URL: <https://habr.com/ru/post/166693/> (Дата обращения 05.05.2022)

23) CYBERLENINKA [Электронный ресурс] ИССЛЕДОВАНИЕ
АЛГОРИТМА SLAM РОБОТОВ НА ОСНОВЕ ОПТИМИЗАЦИИ ГРАФОВ

URL: <https://cyberleninka.ru/article/n/issledovanie-algoritma-slam-robotov-na-osnove-optimizatsii-grafov> (Дата обращения 05.05.2022)

Приложение

Листинг

orb_slam2_mono.launch

```
<launch>

<node name="orb_slam2_mono" pkg="orb_slam2_ros"
type="orb_slam2_ros_mono" output="screen">
<remap from="/camera/image_raw" to="/camera/rgb/image_raw" />
<param name="publish_pointcloud" type="bool" value="true" />
<param name="publish_pose" type="bool" value="true" />
<param name="localize_only" type="bool" value="false" />
<param name="reset_map" type="bool" value="false" />
<!-- static parameters -->
<param name="load_map" type="bool" value="false" />
<param name="map_file" type="string" value="map.bin" />
<param name="voc_file" type="string" value="$(find
orb_slam2_ros)/orb_slam2/Vocabulary/ORBvoc.txt" />
<param name="pointcloud_frame_id" type="string" value="map" />
<param name="camera_frame_id" type="string" value="camera_link" />
<param name="min_num_kf_in_map" type="int" value="5" />
<!-- ORB parameters -->
<param name="/ORBextractor/nFeatures" type="int" value="2000" />
<param name="/ORBextractor/scaleFactor" type="double" value="1.2" />
<param name="/ORBextractor/nLevels" type="int" value="8" />
<param name="/ORBextractor/iniThFAST" type="int" value="20" />
<param name="/ORBextractor/minThFAST" type="int" value="7" />
21
<!-- Camera parameters -->
<!-- Camera frames per second -->
<param name="camera_fps" type="int" value="30" />
<!-- Color order of the images (0: BGR, 1: RGB. It is ignored if
images are grayscale) -->
<param name="camera_rgb_encoding" type="bool" value="true" />
<!-- Camera calibration parameters -->
```

```

<!--If the node should wait for a camera_info topic to take the camera
calibration data-->
<param name="load_calibration_from_cam" type="bool" value="false" />
<!-- Camera calibration and distortion parameters (OpenCV) -->
<param name="camera_fx" type="double" value="632.7927856445312" />
<param name="camera_fy" type="double" value="626.8605346679688" />
<param name="camera_cx" type="double" value="311.43603515625" />
<param name="camera_cy" type="double" value="248.0950164794922" />
<!-- Camera calibration and distortion parameters (OpenCV) -->
<param name="camera_k1" type="double" value="-0.09097914397716522" />
<param name="camera_k2" type="double" value="0.06503549218177795" />
<param name="camera_p1" type="double" value="0.000849052332341671" />
<param name="camera_p2" type="double" value="0.001785792293958366" />
<param name="camera_k3" type="double" value="0.0" />
</node>
</launch>

jetbot_teleop_key.py
import rospy

from geometry_msgs.msg import Twist

import sys, select, os

if os.name == 'nt':
import msvcrt, time
else:
import tty, termios

22

BURGER_MAX_LIN_VEL = 0.22
BURGER_MAX_ANG_VEL = 2.84
WAFFLE_MAX_LIN_VEL = 0.26
WAFFLE_MAX_ANG_VEL = 1.82
LIN_VEL_STEP_SIZE = 0.01
ANG_VEL_STEP_SIZE = 0.1

msg = ""
-----

Moving around:

```

```

w
a s d
x
w/x : increase/decrease linear velocity (Burger : ~ 0.22, Waffle and
Waffle Pi : ~ 0.26)
a/d : increase/decrease angular velocity (Burger : ~ 2.84, Waffle and
Waffle Pi : ~ 1.82)
space key, s : force stop
CTRL-C to quit
"""
e = ""
Communications Failed
"""
def getKey():
    if os.name == 'nt':
        23
        timeout = 0.1
        startTime = time.time()
        while(1):
            if msvcrt.kbhit():
                if sys.version_info[0] >= 3:
                    return msvcrt.getch().decode()
                else:
                    return msvcrt.getch()
            elif time.time() - startTime > timeout:
                return ''
            tty.setraw(sys.stdin.fileno())
            rlist, _, _ = select.select([sys.stdin], [], [], 0.1)
            if rlist:
                key = sys.stdin.read(1)
            else:
                key = ''
            termios.tcsetattr(sys.stdin, termios.TCSADRAIN, settings)
        return key

```

```

def vels(target_linear_vel, target_angular_vel):
    return "currently:\tlinear vel %s\t angular vel %s " %
        (target_linear_vel,target_angular_vel)

def makeSimpleProfile(output, input, slop):
    if input > output:
        output = min( input, output + slop )
    elif input < output:
        output = max( input, output - slop )
    else:
        output = input
24
    return output

def constrain(input, low, high):
    if input < low:
        input = low
    elif input > high:
        input = high
    else:
        input = input
    return input

def checkLinearLimitVelocity(vel):
    if jetbot_model == "burger":
        vel = constrain(vel, -BURGER_MAX_LIN_VEL, BURGER_MAX_LIN_VEL)
    elif turtlebot3_model == "waffle" or turtlebot3_model == "waffle_pi":
        vel = constrain(vel, -WAFFLE_MAX_LIN_VEL, WAFFLE_MAX_LIN_VEL)
    else:
        vel = constrain(vel, -BURGER_MAX_LIN_VEL, BURGER_MAX_LIN_VEL)
    return vel

def checkAngularLimitVelocity(vel):
    if jetbot_model == "burger":
        vel = constrain(vel, -BURGER_MAX_ANG_VEL, BURGER_MAX_ANG_VEL)
    elif jetbot_model == "waffle" or turtlebot3_model == "waffle_pi":
        vel = constrain(vel, -WAFFLE_MAX_ANG_VEL, WAFFLE_MAX_ANG_VEL)
    else:

```

```

vel = constrain(vel, -BURGER_MAX_ANG_VEL, BURGER_MAX_ANG_VEL)

return vel

if __name__=="__main__":
    if os.name != 'nt':
        settings = termios.tcgetattr(sys.stdin)

    rospy.init_node('jetbot_teleop')
    pub = rospy.Publisher('cmd_vel', Twist, queue_size=10)
    jetbot_model = rospy.get_param("model", "burger")
    status = 0

    target_linear_vel = 0.0
    target_angular_vel = 0.0
    control_linear_vel = 0.0
    control_angular_vel = 0.0

    try:
        print(msg)
        while not rospy.is_shutdown():
            key = getKey()

            if key == 'w' :

                target_linear_vel = checkLinearLimitVelocity(target_linear_vel +
LIN_VEL_STEP_SIZE)

                status = status + 1

                print(vels(target_linear_vel,target_angular_vel))

            elif key == 'x' :

                target_linear_vel = checkLinearLimitVelocity(target_linear_vel -
LIN_VEL_STEP_SIZE)

                status = status + 1

                print(vels(target_linear_vel,target_angular_vel))

            elif key == 'a' :

                target_angular_vel = checkAngularLimitVelocity(target_angular_vel +
ANG_VEL_STEP_SIZE)

                status = status + 1

                print(vels(target_linear_vel,target_angular_vel))

            elif key == 'd' :

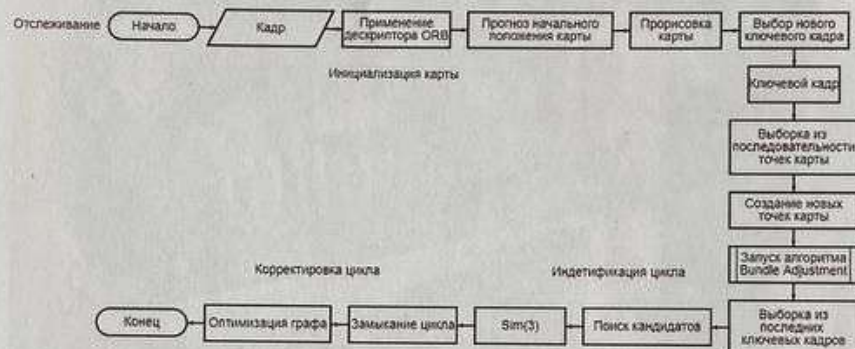
                target_angular_vel = checkAngularLimitVelocity(target_angular_vel -
ANG_VEL_STEP_SIZE)

```

```

status = status + 1
print(vels(target_linear_vel,target_angular_vel))
elif key == ' ' or key == 's' :
    target_linear_vel = 0.0
    control_linear_vel = 0.0
    target_angular_vel = 0.0
    control_angular_vel = 0.0
    print(vels(target_linear_vel, target_angular_vel))
else:
    if (key == '\x03'):
        break
    if status == 20 :
        print(msg)
        status = 0
        twist = Twist()
        control_linear_vel = makeSimpleProfile(control_linear_vel,
        target_linear_vel, (LIN_VEL_STEP_SIZE/2.0))
        twist.linear.x = control_linear_vel; twist.linear.y = 0.0;
        twist.linear.z = 0.0
        control_angular_vel = makeSimpleProfile(control_angular_vel,
        target_angular_vel, (ANG_VEL_STEP_SIZE/2.0))
        twist.angular.x = 0.0; twist.angular.y = 0.0; twist.angular.z =
        control_angular_vel
        pub.publish(twist)
    except:
        print(e)
    finally:
        twist = Twist()
        twist.linear.x = 0.0; twist.linear.y = 0.0; twist.linear.z = 0.0
        twist.angular.x = 0.0; twist.angular.y = 0.0; twist.angular.z = 0.0
        pub.publish(twist)
    if os.name != 'nt':
        termios.tcsetattr(sys.stdin, termios.TCSADRAIN, settings)

```



ИИИ
21K8912

РТУ МИРЭА.03XXXX.ДПГ4-15.03.06

| Изм. | Лист | № докум. | Подпись | Дата |
|----------|------|----------------|--------------------|----------|
| Разраб. | | Санин Н.М. | <i>[Signature]</i> | 16.06.22 |
| Руков. | | Кучерский Р.В. | <i>[Signature]</i> | 16.06.22 |
| | | | | |
| | | | | |
| Зав. каф | | Романов М.П. | <i>[Signature]</i> | 16.06.22 |

Блок-схема макета
разработанного
программного обеспечения

| Лит. | Масса | Масштаб |
|------|-------|----------|
| | | |
| Лист | 1 | Листов 6 |

Кафедра проблем
управления



| | |
|---------------------|--|
| Наименование | 8MP HD resolution 160° FOV wide angle camera |
| Разрешение матрицы | 8 мегапикселей |
| Поле зрения (FoV) | 160° |
| Тип матрицы | Sony IMX219 Color CMOS 8-megapixel |
| Количество пикселей | 3280 x 2464 full FOV, 0.1fps to 15fps |
| Разъем оптики | telephoto to fisheye |
| Размер | 36 x 36 мм |

| | | | | | | | | | |
|----------------|------|----------------|--------------------|----------|---|--|-------------------------------|----------|---------|
| ИИИ 21K8912 | | | | | РТУ МИРЭА.03XXXX.ДДПГ4-15.03.06 | | | | |
| | | | | | Технические характеристики камеры мобильного робота | | Лит. | Масса | Масштаб |
| Изм. | Лист | № докум. | Подпись | Дата | | | | | |
| Разраб. | | Санин Н.М. | <i>[Signature]</i> | 16.06.22 | | | | | |
| Руков. | | Кучерский Р.В. | <i>[Signature]</i> | 16.06.22 | | | | | |
| | | | | | | | Лист 2 | Листов 6 | |
| | | | | | | | Кафедра проблем управления | | |
| Зав. каф | | Романов М.П. | <i>[Signature]</i> | 16.06.22 | | | | | |



ИИИ
21K8912

РТУ МИРЭА.03XXXX.ДДПГ4-15.03.06

| Изм. | Лист | № докум. | Подпись | Дата |
|----------|------|----------------|--------------------|----------|
| Разраб. | | Санин Н.М. | <i>[Signature]</i> | 16.06.22 |
| Руков. | | Кучерский Р.В. | <i>[Signature]</i> | 16.06.22 |
| | | | | |
| | | | | |
| Зав. каф | | Романов М.П. | <i>[Signature]</i> | 16.06.22 |

Блок-схема алгоритма
локальной навигации и
картографирования

| Лит. | Масса | Масштаб |
|------|-------|----------|
| | | |
| Лист | 3 | Листов 6 |

Кафедра проблем
управления





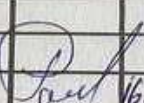
| | |
|-----------------------|--|
| Наименование | JetBot |
| Операционная система | Ubuntu 18.04 LTS |
| Язык программирования | Python |
| Камера | 8MP HD resolution 160° FOV wide angle camera |
| Дисплей | OLED 0.91" 128×32 pixels |
| Протоколы связи | WiFi, Bluetooth 4.2 |
| Тип движка | Differential 2WD |
| Объем аккумулятора | 12.6V, 18650 battery × 3 |
| Тип мотора | TT motor Reduction rate 1:48 Idle speed 240RPM |

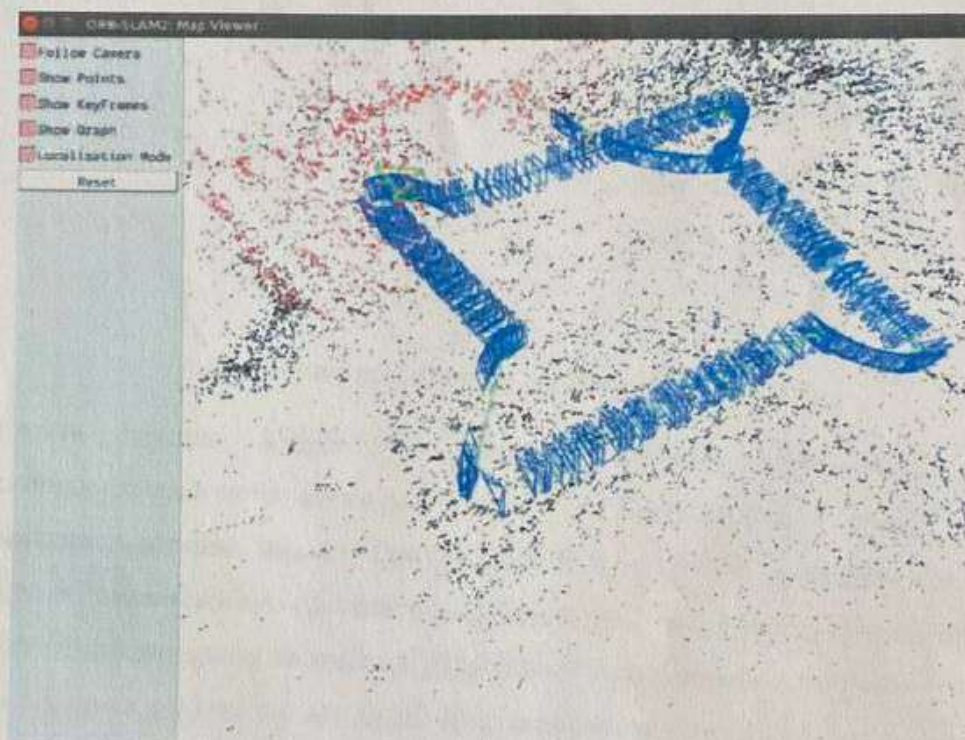
| | | | | | | | | |
|--|------|----------------|--------------------|----------|--------------------------------|-------|---------|--|
| ИИИ 21K8912 | | | | | РТУ МИРЭА.03XXXX.ДПГ4-15.03.06 | | | |
| Технические характеристики мобильного робота | | | | | Лит. | Масса | Масштаб | |
| | | | | | | | | |
| Изм. | Лист | № докум. | Подпись | Дата | Лист 4 Листов 6 | | | |
| Разраб. | | Санин Н.М. | <i>[Signature]</i> | 16.06.22 | Кафедра проблем управления | | | |
| Руков. | | Кучерский Р.В. | <i>[Signature]</i> | 16.06.22 | | | | |
| | | | | | | | | |
| Зав. каф | | Романов М.П. | <i>[Signature]</i> | 16.06.22 | | | | |

| Последовательность | Длина, м | СКО траекторий, м | | | |
|---------------------|----------|-------------------|-------------------|-----------------------|-------------------|
| | | Gmapping | Hector-SLAM | Cartographer (онлайн) | RTAB-MAP |
| 2011-01-19-07-49-38 | 68 | 0.216 ± 0.012 | 1.280 ± 0.640 | 0.188 ± 0.023 | 0.191 ± 0.001 |
| 2011-01-20-07-18-45 | 76 | 0.219 ± 0.012 | 0.245 ± 0.045 | 0.219 ± 0.002 | 0.221 ± 0.001 |
| 2011-01-21-09-01-36 | 87 | 0.212 ± 0.029 | 0.242 ± 0.005 | 0.217 ± 0.003 | 0.205 ± 0.001 |
| 2011-01-24-06-18-27 | 87 | 0.290 ± 0.035 | 0.245 ± 0.006 | 0.217 ± 0.001 | 0.217 ± 0.001 |
| 2011-01-25-06-29-26 | 109 | 0.208 ± 0.008 | 0.260 ± 0.005 | 0.232 ± 0.001 | 0.232 ± 0.002 |
| 2011-01-27-07-49-54 | 94 | 0.266 ± 0.012 | 0.620 ± 0.030 | 0.266 ± 0.004 | – |
| 2011-01-28-06-37-23 | 145 | 2.388 ± 1.949 | 2.280 ± 0.750 | 0.360 ± 0.069 | 0.354 ± 0.004 |
| 2011-03-11-06-48-23 | 245 | 0.365 ± 0.208 | 0.860 ± 0.390 | 1.152 ± 0.601 | 1.348 ± 0.001 |
| 2011-03-18-06-22-35 | 80 | 0.145 ± 0.023 | 0.103 ± 0.008 | 0.145 ± 0.021 | – |
| 2011-04-06-07-04-17 | 95 | 0.190 ± 0.002 | 0.343 ± 0.025 | 0.201 ± 0.002 | – |
| 2011-10-20-11-38-39 | 264 | 0.352 ± 0.003 | 5.486 ± 2.603 | 2.217 ± 0.021 | – |

ИИИ
21K8912

РТУ МИРЭА.03XXXX.ДДПГ4-15.03.06

| | | | | | Лит. | | | Масса | Масштаб |
|----------|------|----------------|---|----------|--|--|--|-------|---------|
| Изм. | Лист | № докум. | Подпись | Дата | Результат сравнительного анализа алгоритмов SLAM | | | | |
| Разраб. | | Санин Н.М. |  | 16.06.22 | | | | | |
| Руков. | | Кучерский Р.В. |  | 16.06.22 | | | | | |
| | | | | | | | | | |
| | | | | | | | | | |
| | | | | | | | | | |
| Зав. каф | | Романов М.П. |  | 16.06.22 | | | | | |
| | | | | | Лист 5 Листов 6 | | | | |
| | | | | | Кафедра проблем управления | | | | |



ИИИ
21K8912

РТУ МИРЭА.03XXXX.ДДПГ4-15.03.06

| | | | | | Лит. | Масса | Масштаб |
|----------|------|----------------|--------------------|----------|---|-------|---------|
| Изм. | Лист | № докум. | Подпись | Дата | Результаты работоспособности макета разработанного ПО | | |
| Разраб. | | Санин Н.М. | <i>[Signature]</i> | 16.06.22 | | | |
| Руков. | | Кучерский Р.В. | <i>[Signature]</i> | 16.06.22 | Лист 6 Листов 6 | | |
| | | | | | | | |
| | | | | | Кафедра проблем управления | | |
| Зав. каф | | Романов М.П. | <i>[Signature]</i> | 16.06.22 | | | |