Министерство образования и науки Российской Федерации

Московский авиационный институт (национальный исследовательский университет)

«Создание роботизированной платформы автономного позиционирования и навигации»

Автор Седов Михаил Артёмович

Группа: М8О-116БВ-24

Руководитель Крылов Евгений Евгеньевич

Инженер внедрения ООО «Цифровая индустриальная платформа»

Г. Москва

2024–2025 учебный год

СОДЕРЖАНИЕ

[ВВЕДЕНИЕ 2](#_Toc183798428)

[Актуальность 2](#_Toc183798429)

[Объект 2](#_Toc183798430)

[Предмет 2](#_Toc183798431)

[Цель работы 2](#_Toc183798432)

[Задачи 2](#_Toc183798433)

[Описание ROS 1 4](#_Toc183798434)

[Дифференциальный привод с гусеницами 5](#_Toc183798435)

[Теория навигации 7](#_Toc183798436)

[Лидар 9](#_Toc183798437)

[Сборка 11](#_Toc183798438)

[Программирование 13](#_Toc183798439)

[Внедрение 19](#_Toc183798440)

[Заключение 21](#_Toc183798441)

[СПИСОК ИСПОЛЬЗОВАННОЙ ЛИТЕРАТУРЫ 23](#_Toc183798442)

[Приложения 24](#_Toc183798443)

[Плата управления приводами 24](#_Toc183798444)

[Архитектура системы 26](#_Toc183798445)

[Описание алгоритма gmapping 27](#_Toc183798446)

# ВВЕДЕНИЕ

## Актуальность

Умные носилки должны будут выполнять часть функций медперсонала: перевозку больных по медицинским заведениям, помощь посетителям в ориентировании по медицинским учреждениям. Ориентироваться по сооружениям они будут по встроенным в них гироскопам, картам, камере, помогающей объезжать людей.

## Объект

Роботизированная платформа автономного позиционирования и навигации под управление Robotic Operating System (ROS) на базе микрокомпьютера Raspberry Pi.

## Предмет

Работа алгоритма SLAM (одновременное позиционирование и построение карты местности); работа лидара; Взаимодействие платы микрокомпьютера и подключенных устройств; основы программирования для ROS; организация удалённого взаимодействия управляющего ПК с Ubuntu и робота с ROS

## Цель работы

Сборка, программирование, тестирование и внедрение робота, созданного на базе Raspberry Pi, с помощью ROS и Ubuntu.

## Задачи

1. Сборка:
   1. Электрические соединения:
      1. Электродвигатели, 2 шт.
      2. Плата контролеров + гироскоп
      3. Raspberry Pi
      4. Лидар
      5. Дисплей
      6. Видеокамера
      7. Аккумулятор
   2. Механические узлы
      1. Шасси
      2. Зубчатые и опорные колёса
      3. Гусеницы
      4. Крепёж для электрических узлов
2. Программирование
   * 1. Инициализация операционных систем Raspbian и ROS на плате Raspberry Pi.
     2. Развёртывание операционной системы Ubuntu 16на ПК
     3. Развёртывание операционной системы ROS и утилит для сборки программного кода робота в среде Ubuntu
     4. Компиляция кода робота и перенос кода на Raspberry Pi
     5. Соединение операционных систем ROS между ПК и Raspberry Pi посредством протокола SSH
3. Тестирование
   * 1. Тестирование кода робота в эмуляторе под Ubuntu
     2. Инициализация ROS на роботе по средствам SSH
     3. Калибровка гироскопа
     4. Построение и сохранение карты местности в памяти робота с помощью лидара и ручного управления движением робота
     5. Запуск на роботе команд автономной навигации по заданным точкам на карте с объездом ненанесённых на карту препятствий
4. Внедрение

# Описание ROS 1

ROS (Robot Operation System) — это операционная система для роботов. Она обеспечивает всю необходимую функциональность для распределённой работы всех узлов робота. На самом деле ROS — это библиотека, надстройка поверх компьютерной операционной системы. ROS предоставляет стандартные возможности операционной системы, такие как аппаратная абстракция, низкоуровневое управление устройствами, реализация часто используемых функций, передача сообщений между процессами и управление пакетами.

ROS содержит множество пакетов для создания виртуального робота и симулирования его поведения — например, стек пакетов gazebo\_ros\_pkgs. C помощью Gazebo вы сможете даже симулировать знаменитого робота Atlas от Boston Dynamics у себя на компьютере.

# Дифференциальный привод с гусеницами

Дифференциальный привод с гусеницами (танковое шасси) — версия привода skid-steer с гусеницами вместо дополнительных колёс. Каждая гусеница и сторона робота контролируется одним мотором.

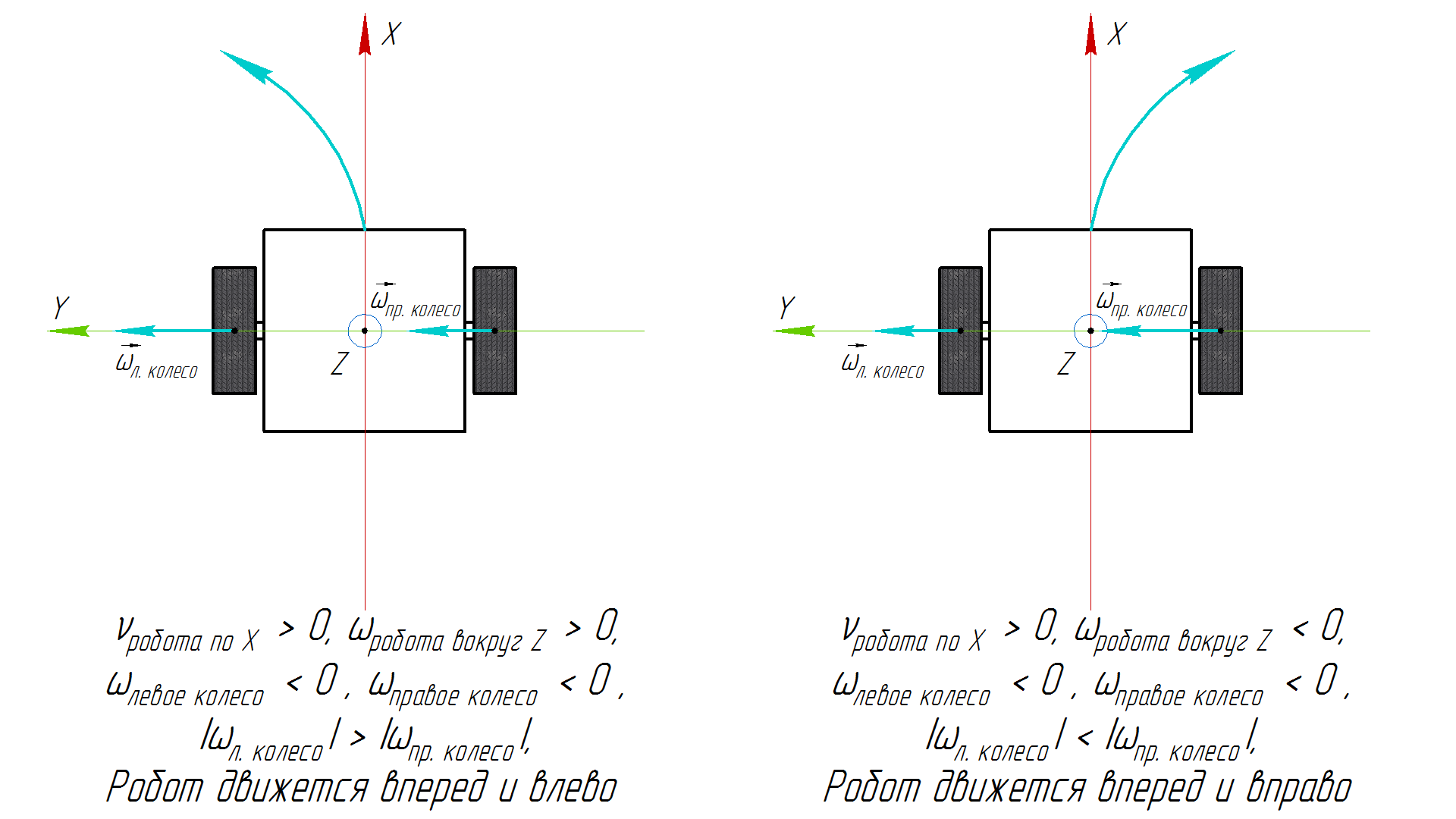
Можно интерпретировать этот привод как двухколёсный дифференциальный, где колесо имеет некруглую форму и увеличенную длину окружности (Рисунок 1). Или как привод skid-steer с бесконечным количеством колёс на определённой длине.

Рисунок 1. Принцип действия дифференциального привада с двумя колёсами (гусеницами)

Движение привода с гусеницами неголономно. Принцип движения тут такой же, как и у привода skid-steer. Только для обработки перемещения робота используются не угловые скорости колёс, а скорости гусениц.

Особенности шасси:

• Танковое шасси обладает самыми высокими эксплуатационными характеристиками на пересечённой местности благодаря форме гусениц и сцеплению с землёй.

• Усложнённая механика. В конструкции танкового шасси множество непростых деталей: части трака, натяжители, опорные ролики и т. д.

• Получить одометрию ещё сложнее, чем при использовании привода skid-steer. При движении танкового шасси тоже происходят проскальзывания гусениц и заносы (особенно это заметно, когда робот-танк вращается на месте на ровной поверхности). Однако ввиду наличия всего двух датчиков вращения программно компенсировать ошибки одометрии очень тяжело. При движении по ровной поверхности одометрия неточная, а на пересечённой местности датчики вращения становятся практически бесполезны, и для одометрии понадобятся другие источники.

# Теория навигации

Автономная навигация робота строится на трёх фундаментальных принципах:

1. Построение карт — Mapping.
2. Локализация в пространстве — Localization.
3. Планирование пути — Path planning.

Их задачи:

1. Задача маппинга состоит в том, чтобы ответить на вопрос робота: «Как выглядит окружающее меня пространство?» Во время картографирования данные с различных датчиков передаются роботу. На основе этих данных робот строит карту окружающего мира (map) в понятном для себя представлении — топологическом или метрическом.
2. Задача локализации состоит в том, чтобы ответить на вопрос робота: «Где я нахожусь в окружающем меня мире?» Во время локализации робот определяет своё положение относительно карты (map), которая может быть уже известна роботу или строиться в режиме реального времени. При локализации на заранее известной карте робот должен уметь определять своё положение, где бы он не находился.
3. Задача планирования пути состоит в том, чтобы ответить на вопрос робота: «Как я могу добраться до определённой точки на карте?» Целевая точка на карте может быть установлена оператором робота или самим роботом. Он должен уметь cамостоятельно прокладывать траекторию движения к целевой точке на карте и добираться до этой точки. Кроме этого, траектория движения должна быть оптимальна и безопасна для робота.

Различные комбинации этих трёх процессов позволяют мобильному роботу решать разнообразные навигационные задачи (Рисунок 2):

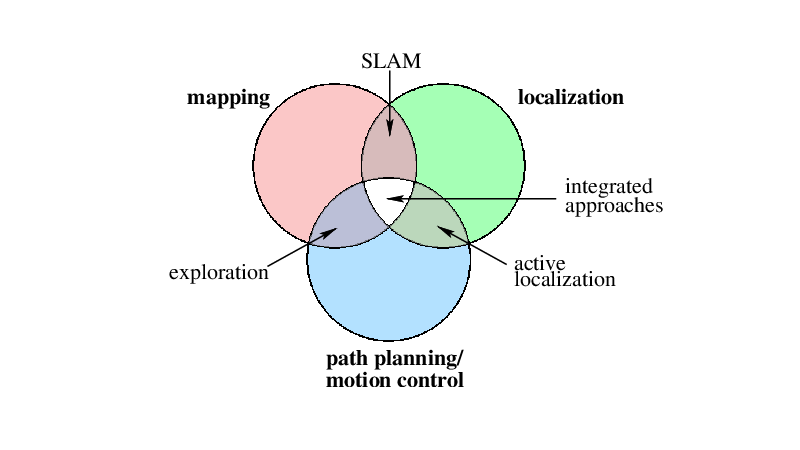
. 

Рисунок 2. Принцип действия трёх процессов: маппинга локализации планирования пути

Что из всего этого будем использовать мы? Для начала мы составим методом SLAM глобальную карту помещения, где будет работать наш робот.

Затем мы дадим роботу эту карту помещения и будем управлять им, задавая целевые точки (Goal points) на ней. До этих точек робот должен будет добираться самостоятельно, объезжая все препятствия на пути и при этом не теряя себя на карте. Таким образом, мы будем использовать метод Active localisation.

# Лидар

**Устройство и принцип работы**  
Лидар (Light Detection and Ranging) использует лазерные импульсы для измерения расстояния до объектов и создания их точных 3D-моделей. Устройство состоит из следующих ключевых компонентов:

* **Лазерный источник**: генерирует импульсы света.
* **Сканер и оптика**: направляют лучи и принимают их отражение от объектов, обеспечивая охват до 360° в современных системах.
* **Процессор**: обрабатывает данные о времени полета луча (Time-of-Flight) и преобразует их в облако точек для 3D-визуализации.
* **Инерциальный блок и GPS**: фиксируют положение лидара для мобильных платформ.

**Преимущества лидара перед аналогами**

1. **Высокая точность**: Лидары создают карты с детализацией до нескольких сантиметров, что особенно важно для робототехники и беспилотных автомобилей.
2. **Работа в любых условиях освещения**: В отличие от камер, лидар работает независимо от уровня света, что делает его эффективным ночью и в условиях плохой видимости.
3. **Широкий угол обзора**: Некоторые системы обеспечивают обзор в 360°, что превосходит традиционные радары【21】【22】.
4. **Скорость обработки**: Высокопроизводительные процессоры позволяют обрабатывать миллионы импульсов в секунду, что делает лидары идеальными для систем реального времени【19】【20】.
5. **Универсальность**: Используются в навигации, геодезии, строительстве, медицине, а также в системах дополненной реальности и мобильных устройствах【21】.

**Сравнение с другими технологиями**

* **Радар**: Использует радиоволны и дешевле, но имеет меньшую точность и чувствительность к мелким объектам.
* **Камеры**: Дешевле лидаров, но зависят от освещения и сложнее обрабатывают данные для 3D-картирования【22】.



Рисунок 3. Лидар

# Сборка

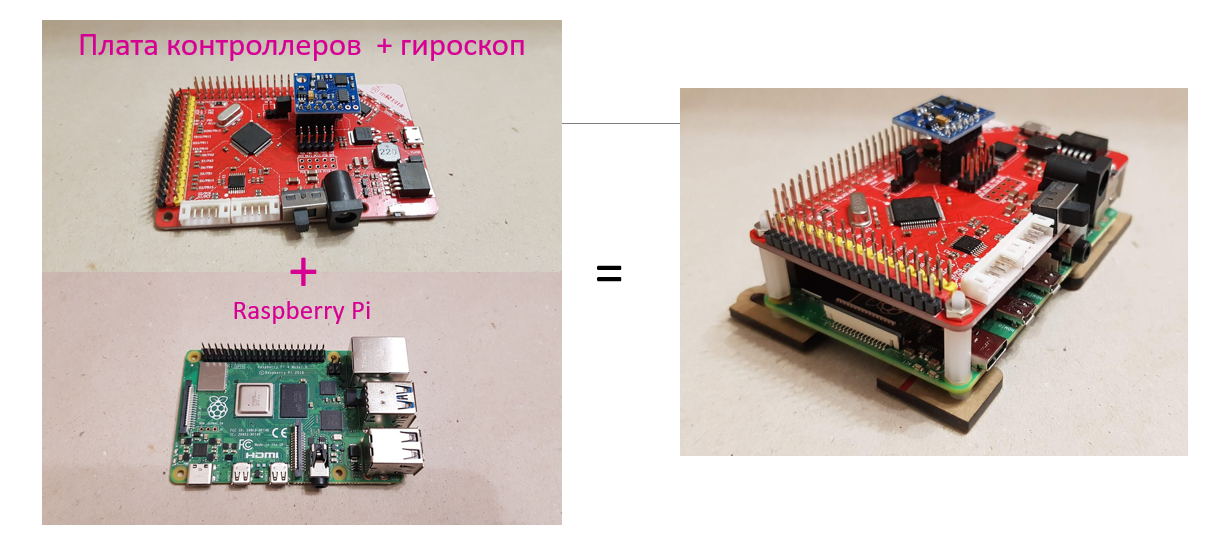


Рисунок 4. Подключение платы контролера к Raspberry Pi

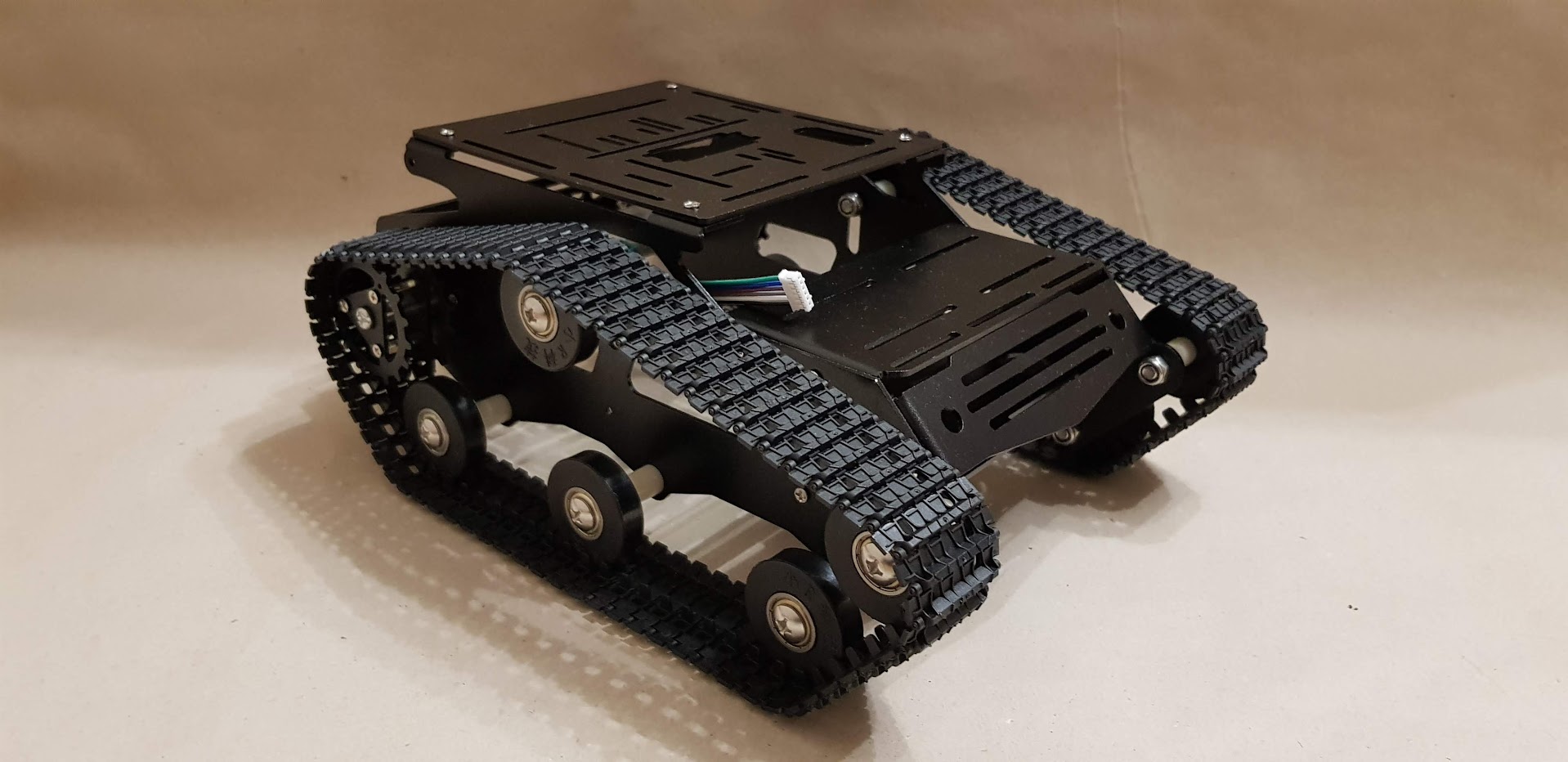


Рисунок 5. Шасси в сборе



Рисунок 6. Робот в сборе

# Программирование

Развернул виртуальную машину VMware с операционной системой Ubuntu 16.04.

Изображение выглядит как текст, монитор, снимок экрана, экран

Автоматически созданное описание

Рисунок 7. Вход в Ubuntu

Развернул операционную систему ROS Kinetic с помощью команды

sudo apt-get install ros-kinetic-full

Выполнил сборку кода робота с помощью утилиты catkin\_make

catkin\_make

source devel/setup .bash

Перенёс готовый код на карту памяти робота

Подключился по SSH к роботу

ssh xrrobot@192.168.1.1

Инициализировал операционную систему ROS командой

roscore

Изображение выглядит как текст

Автоматически созданное описание

Рисунок 8. Инициализация ROS

Тестирование

1. Подключился по SSH к роботу

ssh xrrobot@192.168.1.1

1. Запустил пакет навигации

roslaunch xrrobot bringup.launch

Изображение выглядит как текст

Автоматически созданное описание

Рисунок 9. Показатели работы энкодеров робота после запуска навигации

1. Запустил калибровку гироскопа

rosrun imu\_do\_calib

Изображение выглядит как текст

Автоматически созданное описание

Рисунок 10. Процесс калибровки гироскопа

1. Запустил процесс построения карты

roslaunch xrrobot bring\_slam.launch

roslaunch xrrobot bringup.launch

roslaunch xrrobot lidar\_slam.launch

1. Запустил утилиту визуализации

rosrun rviz rviz

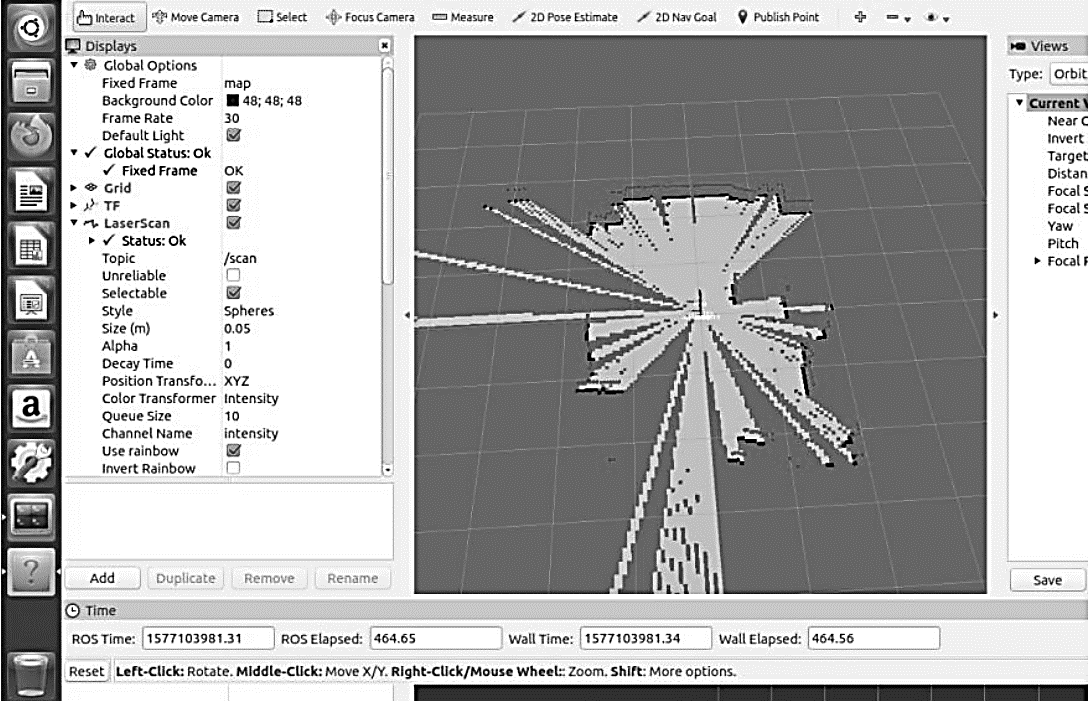


Рисунок 11. Визуализация при построении карты

1. Запустил управление с клавиатуры

rosrun teleop\_twist\_keyboard teleop\_twist\_keyboard.py

Изображение выглядит как текст

Автоматически созданное описание

Рисунок 12. Управление робота с клавиатуры

1. Провёл робота по помещению для построения карты
2. Сохранил карту

roscd xrrobot/maps/

map.sh

1. Запустил автономную навигацию

roslaunch xrrobot bring\_navigate.launch

1. Указал в утилите визуализации начальную точку на карте

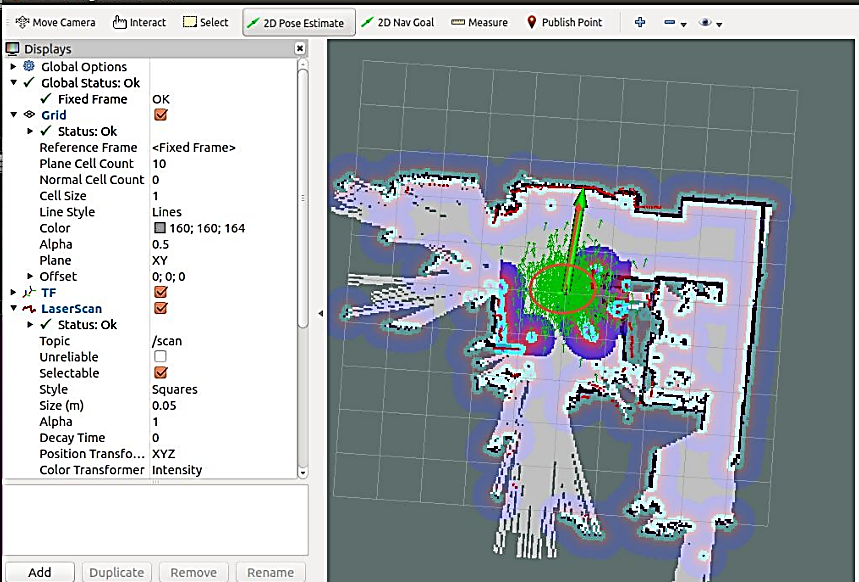


Рисунок 13. Инициация начальной точки робота

Указал в утилите визуализации конечную точку на карте



Рисунок 14. Инициация начальной точки робота

1. Робот самостоятельно проследовал в конечную точку, объезжая препятствия.

# Внедрение

Для успешного внедрения умных носилок в больничную структуру необходимо учесть несколько ключевых этапов:

1. **Техническая подготовка**:
   * Установить систему навигации, используя технологии SLAM и датчики, такие как лидары, для обеспечения автономного передвижения носилок. Эти технологии помогут строить карты помещений, планировать маршруты и обходить препятствия.
   * Оснастить носилки модулями связи (Wi-Fi, Bluetooth) для интеграции с больничной IT-инфраструктурой. Это позволит передавать данные о перемещении и местоположении в реальном времени.
2. **Интеграция в IT-системы**:
   * Объединить носилки с системой управления больницей. Например, носилки могут получать задания из электронных медицинских карт, автоматизируя маршруты доставки пациентов или оборудования.
   * Разработать интерфейс для медицинского персонала, чтобы они могли быстро задавать цели и получать данные о статусе задач.
3. **Тестирование и адаптация**:
   * Проведите пилотное внедрение в нескольких корпусах больницы, чтобы выявить недостатки и скорректировать настройки навигации и маршрутизации.
   * Организуйте обучение медицинского персонала работе с новой системой, включая настройку задач для носилок и мониторинг их выполнения.
4. **Повышение операционной эффективности**:
   * Используйте носилки для выполнения рутинных задач, таких как доставка оборудования и транспортировка пациентов, снижая нагрузку на персонал и минимизируя риски травм.
   * Автоматизация перемещений поможет оптимизировать логистику внутри больницы, сокращая время на поиск и транспортировку.
5. **Мониторинг и масштабирование**:
   * Регулярно собирайте данные о работе носилок для анализа их эффективности. Это позволит оптимизировать их использование и сократить затраты.
   * После успешного пилота постепенно внедряйте технологию в других зданиях и отделениях.

Внедрение умных носилок не только снизит нагрузку на персонал, но и улучшит качество обслуживания пациентов, сделав больницы более безопасными и эффективными.

# Заключение

Результат решения каждой задачи:

1. Был создан каркас робота, с помощью
   1. Электрических соединений:
      1. Электродвигатели, 2 шт.
      2. Плата контролеров + гироскоп
      3. Raspberry Pi
      4. Лидар
      5. Дисплей
      6. Видеокамера
      7. Аккумулятор
   2. Механических узлов
      1. Шасси
      2. Зубчатые и опорные колёса
      3. Гусеницы
      4. Крепёж для электрических узлов
2. Программирование
   1. Произведена инициализация операционных систем Raspbian и ROS на плате Raspberry Pi.
   2. Произведено развёртывание операционной системы Ubuntu 16на ПК
   3. Произведено развёртывание операционной системы ROS и утилит для сборки программного кода робота в среде Ubuntu
   4. Произведена компиляция кода робота и перенос кода на Raspberry Pi
   5. Произведено соединение операционных систем ROS между ПК и Raspberry Pi посредством протокола SSH
3. Тестирование
   1. Произведено тестирование кода робота в эмуляторе под Ubuntu и на роботе
   2. Произведена инициализация ROS на роботе по средствам SSH
   3. Произведена калибровка гироскопа
   4. Произведено построение и сохранение карты местности в памяти робота с помощью лидара и ручного управления движением робота
   5. Произведен запуск на роботе команд автономной навигации по заданным точкам на карте с объездом ненанесённых на карту препятствий

# СПИСОК ИСПОЛЬЗОВАННОЙ ЛИТЕРАТУРЫ

1. https://habr.com/ru/post/128046/ - Основы работы с Robotic Operating System 2: скажи миру «Hello, world!»
2. https://habr.com/ru/post/128024/ - Основы работы с Robotic Operating System
3. https://www.xiaorgeek.com/Study/Study/catalog/cid/35 - Инструкция по сборке и основ программирования робота
4. https://www.youtube.com/watch?v=gxhTOWCMVvA - Введение в Robot Operating System
5. <https://amperka.ru/blogs/projects/abot-robot-part-2#navigation-theory> - Как сделать робота на ROS своими руками

# Приложения

## Плата управления приводами

Основные элементы платы управления приводами показаны на следующем рисунке.

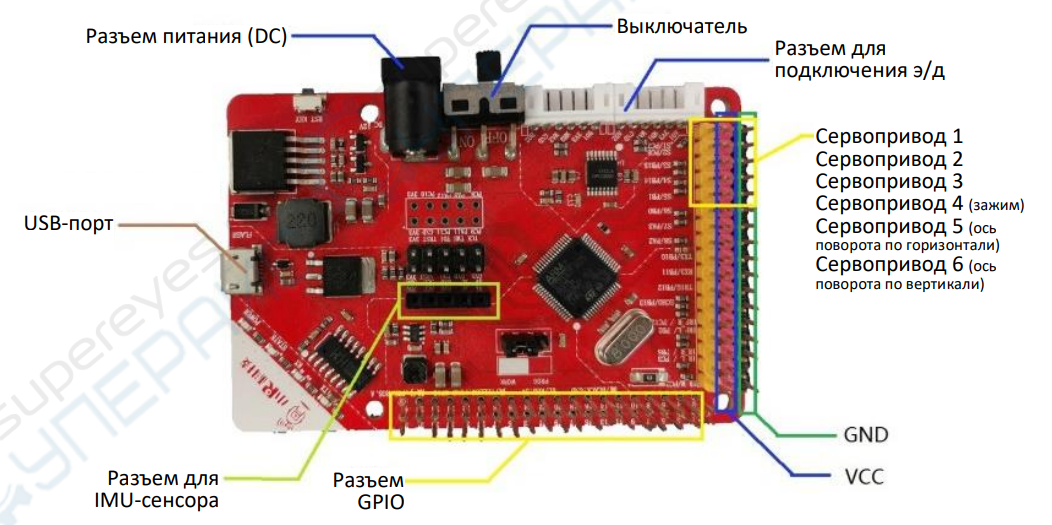
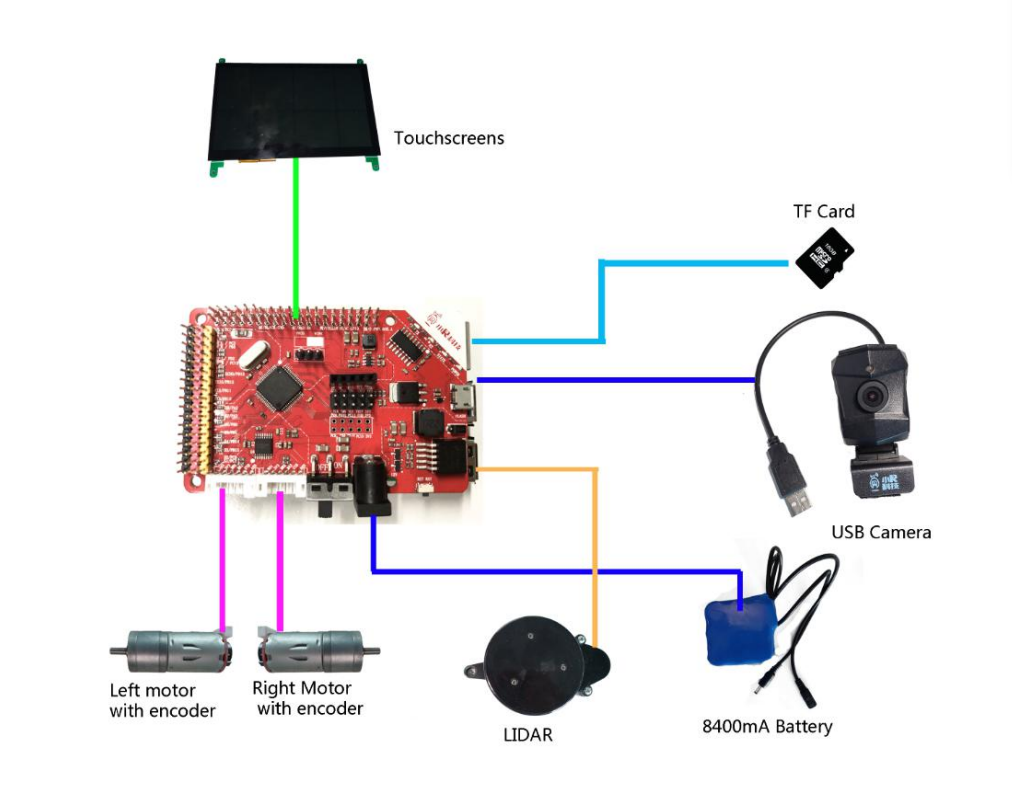


Рисунок 15. Внешний вид платы управления приводами



Сенсорный экран

Карта памяти

USB-камера

Аккумулятор, 8400 мА

Лидар

Левый и правый двигатели   
с энкодерами

Рисунок 16. Схема подключения платы управления приводами

PWR.ROS.A — это плата для управления приводами робота, соединяемая с платой Raspberry Pi 4B. Ее основные функции заключаются в следующем:

1. Стабилизация напряжения: данная плата обеспечивает понижение входного напряжения 7–12 В постоянного тока до 5 В или другого напряжения, которое требуется для питания основной платы управления или приводов.
2. Управление электродвигателями: данная плата обеспечивает управление приводами для движения робота вперед и назад, остановки, ускорения, замедления.
3. Управление сервоприводами: данная плата оснащена 8-канальным выходным разъемом для подключения к сервоприводам. При питании сервоприводов от платы последняя подходит для одновременного ШИМ-управления восемью сервоприводами с напряжением питания 5 В и максимальным током не более 100 мА.
4. Подключение датчиков: неиспользуемые разъемы GPIO на данной плате подключаются напрямую к плате Raspberry Pi снизу (контакты обозначены желтым цветом).
5. Прием данных с датчиков, включая частоту вращения электродвигателя, показания гироскопа и другие основные параметры.

## Архитектура системы

На следующей иллюстрации показана блок-схема работы алгоритма навигации.

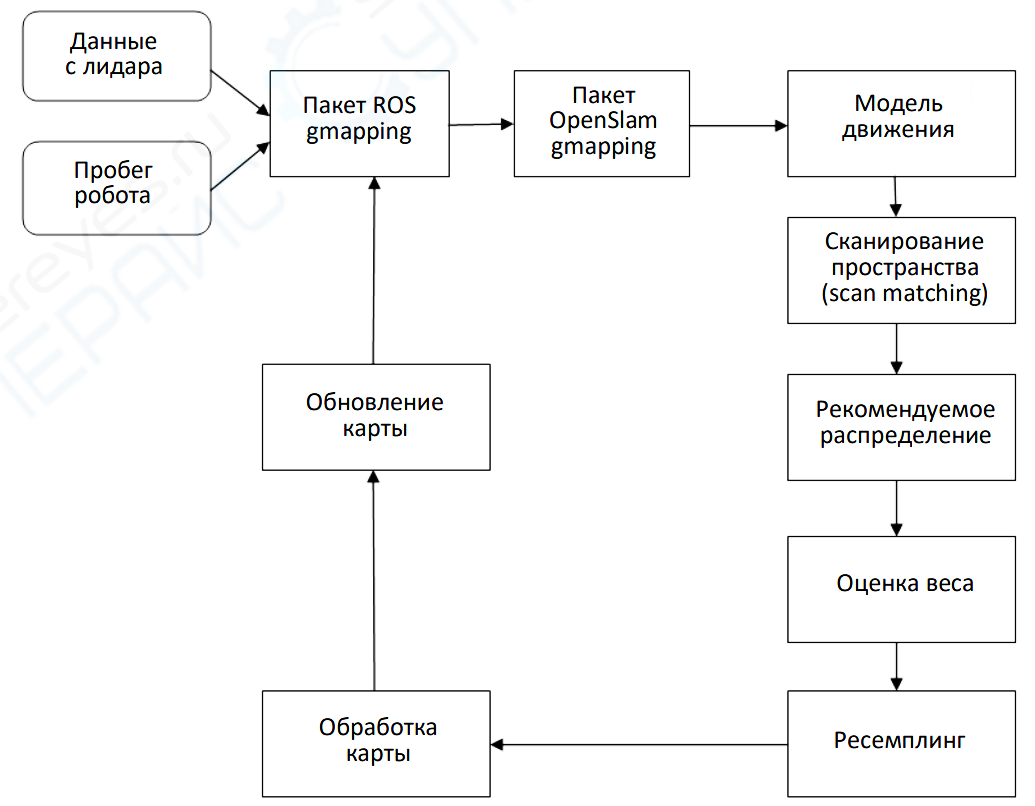


Рисунок 17. Архитектура системы

## Описание алгоритма gmapping

Gmapping — это лазерный алгоритм SLAM, который был интегрирован в ROS, и это наиболее часто используемый алгоритм SLAM среди мобильных роботов. Этот алгоритм был предложен Grisetti и др. в качестве метода SLAM фильтрации частиц, основанного на Rao-Blackwellized. Алгоритм, основанный на фильтре частиц, использует множество взвешенных частиц для представления апостериорной вероятности пути, и каждая частица предлагает важный фактор. Однако для получения хороших результатов обычно требуется большое количество частиц, что увеличивает вычислительную сложность алгоритма.

Процесс также снижает точность алгоритма. Проблема деградации частиц связана с удалением большого количества частиц из набора образцов на этапе повторной выборки. Это произошло потому, что их важность может стать незначительной. Следовательно, это означает, что существует определенная небольшая вероятность того, что правильная гипотетическая частица будет исключена с течением времени. Чтобы избежать деградации частиц, был разработан метод адаптивной повторной выборки.

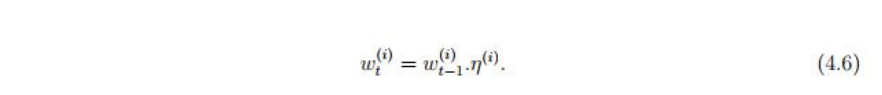
Авторы также предлагают метод расчета точного распределения, учитывающего не только движение платформы робота, но и недавние наблюдения. Предложенное распределение предлагается путем объединения наблюдаемых значений с диапазоном движения. модель. Однако, когда мобильный робот оснащен очень точным датчиком LRF, можно использовать модель этого датчика, поскольку она может достигать функций чрезвычайно высокого правдоподобия. На основе этого авторы интегрируют недавние наблюдения датчика zt и вычисляют распределение Гаусса, чтобы приблизительно эффективно получить следующее поколение распределения частиц.

Авторы интегрируют самое последнее наблюдение датчика 2. Кроме того, они вычисляют приближение Гаусса для эффективного получения следующего поколения частиц. Параметры Гаусса

Изображение выглядит как текст, Шрифт, рукописный текст, белый

Автоматически созданное описание

Авторы интегрируют самое последнее наблюдение датчика 2. Кроме того, они вычисляют приближение Гаусса для эффективного получения следующего поколения частиц. Параметры Гаусса



Где K — количество точек выборки, а µ — нормализованный коэффициент. Используя это распределение, вес i-й частицы определяется как:

Изображение выглядит как зарисовка, белый, дизайн, типография

Автоматически созданное описание

Затем, используя (4.7), они вычисляют эффективное число Neff частиц как критерий, определяющий, когда следует выполнять шаг повторной выборки:

Эта адаптивная повторная выборка снижает неопределенность положения робота на этапах прогнозирования PF. В результате во время процесса сопоставления сканирования требуется меньшее количество частиц из-за низкой неопределенности. В экспериментах gmapping использовалось количество частиц 30, что очень мало по сравнению с обычным методом PF.