«Санкт-Петербургский государственный электротехнический университет «ЛЭТИ» им. В.И.Ульянова (Ленина)»
(СПбГЭТУ «ЛЭТИ»)

Направление	09.03.04 - Программная инженерия
Профиль	Разработка программно-информационных систем
Факультет	КТИ
Кафедра	МО ЭВМ

К защите допустить
Зав. кафедрой Кринкин К.В.

ВЫПУСКНАЯ КВАЛИФИКАЦИОННАЯ РАБОТА БАКАЛАВРА

Тема: ИССЛЕДОВАНИЕ АЛГОРИТМОВ ЛОКАЛЬНОГО ПЛАНИРОВАНИЯ ТРАЕКТОРИЙ КОЛЕСНЫХ РОБОТОВ

Студент(ка) _____ Гришин К. И.
подпись

Руководитель к. т. н., доцент _____ Кринкин К. В.
(Уч. степень, уч. звание) *подпись*

Консультанты асс. каф. МО ЭВМ _____ Чайка К. В.
(Уч. степень, уч. звание) *подпись*

к. т. н., доцент _____ Иванов А. Н.
(Уч. степень, уч. звание) *подпись*

Санкт-Петербург
2022

РЕФЕРАТ

Пояснительная записка 65 стр., 31 рис., 13 табл., 26 ист.

МОБИЛЬНЫЙ РОБОТ, КОЛЕСНЫЙ РОБОТ, ЛОКАЛЬНОЕ ПЛАНИРОВАНИЕ, DYNAMIC WINDOW APPROACH, TIMED-ELASTIC-BAND, TRAJECTORY ROLLOUT, MODEL PREDICTIVE CONTROL.

Цель данной работы: Рассмотреть существующие алгоритмы локального планирования колесных роботов. Определить необходимое оборудование для реализации алгоритмов, а также провести оценку их быстродействия.

Объектом исследования являются алгоритмы локального планирования мобильных роботов

Предмет исследования — скорость генерации и качество локальных маршрутов, составленных алгоритмами локального планирования.

Цель данной работы: Рассмотреть существующие алгоритмы локального планирования колесных роботов. Определить необходимое оборудование для реализации алгоритмов, а также провести оценку их быстродействия.

Для автоматизации движения мобильных роботов необходимо решить три основные задачи: выбор глобального маршрута, определение своего положения в пространстве, выбор локального маршрута и построение траектории движения. В данной работе подробно разобраны различные способы решения последней задачи, а именно локального планирования. Выявлены критерии оценки методов локального планирования. Произведено сравнение различных по своей структуре и идее методов.

ABSTRACT

Mobile robot movement automation consists of three main tasks: Global route selection; robot position determination; local route selection and building a movement trajectory. In this work, various methods for solving the local route selection, otherwise – local planning or motion planning are analyzed in detail. Criteria for evaluating methods of local planning have been defined. Methods different in their structure and idea are compared.

ОПРЕДЕЛЕНИЯ, ОБОЗНАЧЕНИЯ, СОКРАЩЕНИЯ

Локальное планирование — поиск траектории движения в пространстве в непосредственной близости робота путем использования различных бортовых сенсоров.

Мобильный робот — это автоматический механизм, способный перемещаться в окружающем пространстве, не привязанный к одной локации и выполняющий действия согласно интегрированной базе знаний.

Дифференциальный привод — привод в основе которого лежит два мотора, по одному на каждое колесо.

Визуализатор — программное средство, предназначенное для преобразования информации в зрительные образы.

DWA — Dynamic Window Approach

ROS — Robot Operating System

MPC — Model Predictive Control

TEB — Timed-Elastic-Band

TR — Trajectory Rollout

ВВЕДЕНИЕ

Навигация автономных мобильных роботов – быстроразвивающаяся область. За последние десятилетия произошел огромный рост данной отрасли. Автономные роботы или транспортные средства значительно снижают вклад человеческих ошибок. Роботы все чаще работают в закрытых помещениях, предназначенных для людей, в среде, в которой необходимо избегать неожиданные препятствия. Автономные роботы должны безопасно и эффективно перемещаться из точки *А* в точку *Б* с учетом времени, расстояния, энергии и других факторов.

Во время навигации, роботизированные системы запускают процессы, которые включают моделирование окружающей среды и локализацию положения системы в окружающей среде, управление движением, обнаружение и предотвращение препятствий, а также движение в динамических средах [1].

В автономной навигации выделяется три основные задачи: выбор глобального маршрута, определение своего положения, выбор локального маршрута [2]. Первая заключается в том, чтобы построить глобальный маршрут из одной точки в другую на статической карте; Вторая – точное определение местоположения робота в текущей среде. Последняя заключается в том, чтобы правильно вычислить команды, которые направятся роботу, для движения, чтобы отслеживать глобальный путь и избегать столкновений с препятствиями.

В данной работе уделено внимание локальным планировщикам, в частности реализациям и внедрению некоторых планировщиков в Robot Operating System (ROS) для роботов с дифференциальным приводом.

ROS – это фреймворк, широко используемый в сообществе робототехники. Его основная цель – сделать разработку ПО для роботов более гибкой. ROS — это набор инструментов, библиотек и соглашений, используемых для взаимодействия с роботизированными платформами.

В данной работе разобраны все, используемые в ROS, локальные планировщики, а также же те, которые скоро будут добавлены.

В качестве среды для испытаний, использовалось ПО для симуляции пространства и робота Gazebo. Данная платформа крайне популярна в ROS сообществе, имеет поддержку физики и полноценно интегрирована в ROS. Она представляет собой виртуальный мир, в котором расположен робот. Интерфейс взаимодействия с предоставляется ROS будто это настоящий робот.

Цель данной работы: Рассмотреть существующие алгоритмы локального планирования колесных роботов. Определить необходимое оборудование для реализации алгоритмов, а также провести оценку их быстродействия.

Задачи данной работы:

1. Изучение существующих алгоритмов локального планирования для мобильных роботов.
2. Разработка ПО для автономного управления колесным роботом.
3. Определение метрик для сравнения алгоритмов.
4. Определение эффективности применения существующих алгоритмов локального планирования роботов в неизвестной среде.

Объектом исследования являются алгоритмы локального планирования мобильных роботов

Предмет исследования — скорость генерации и качество локальных маршрутов, составленных алгоритмами локального планирования.

Практическая ценность работы: Заранее составленная траектория движения робота не позволяет полностью автоматизировать перемещение. Окружение и препятствия постоянно изменяются, может ухудшаться

видимость или поверхность передвижения, в таком случае необходимо прибегать к методам локального планирования.

Необходимо провести анализ существующих алгоритмов для определения их быстродействия и применимости к маленьким маневренным колесным роботам.

Помимо программных ограничений, также существуют и аппаратные, устанавливаемые сенсоры могут иметь плохое разрешение или шумы, стоит учитывать стоимость оборудования для использования того или иного алгоритма.

В дополнение, готовая реализация алгоритма на реальном роботе позволит более удобно проводить дальнейшие исследования построения глобальных маршрутов и SLAM-алгоритмов.

5 БЕЗОПАСНОСТЬ ЖИЗНЕДЕЯТЕЛЬНОСТИ

Основной задачей данной работы являлось исследование поведения робота при применении различных алгоритмов локального робота в симмуляционной среде. В настоящее время одним из важнейших этапов разработки ПО в робототехнике является проведение экспериментов в симуляторе, поэтому необходимо уделить внимание эргономике интерфейсов взаимодействия в с ПО для навигации и симуляции пространства.

В связи с широким развитием области информационных технологий и масштабом производимой продукции, к программным средствам также применяются стандарты, требующие создания наиболее удобных пользовательских интерфейсов. От пользовательского интерфейса зависят качество и скорость проводимых экспериментов.

5.1 Основные положения об эргономике

Требования по разработке интерфейса человек – система изложены в ГОСТ Р ИСО 9241-100 [26]. Этот документ необходимо использовать при оценке эргономики ПО.

Согласно ГОСТ, эргономика – это наука рассматривающая вопросы взаимодействия человека с другими элементами интерактивной системы.

Основной задачей ГОСТ является введение принципов, рекомендаций и требований для пользовательских интерфейсов, обеспечивающих диалог между пользователем и системой, рассматриваемый как последовательность действий (ввод) и ответных реакций системы (вывод), направленный на достижение цели и выполнение производственной задачи.

Стандарты по эргономике ПО применимы ко всем компонентам системы, включая:

- прикладные программы

- операционные системы
- встроенное ПО
- программные средства проектирования
- вспомогательные технологии

Стоит принять во внимание, что стандарты по эргономике носят больше рекомендательный характер, т. к. описываемые требования к пользовательским интерфейсам не обязательны. Однако соблюдение этих требований позволяет предотвратить возникновение будущих проблем с пригодностью ПО к использованию.

В ГОСТ Р ИСО 9241-100 приведены семь высокоуровневых эргономических принципов для разработки диалога между пользователем и системой:

1. Пригодность для выполнения задачи
2. Информативность
3. Соответствие ожиданиям пользователя
4. Пригодность для обучения
5. Управляемость
6. Устойчивость к ошибкам
7. Пригодность для индивидуализации

Данные принципы направлены на понимание требований в области эргономики и нет строгой необходимости в проверке их выполнения.

5.2 Требования к эргономике используемого ПО

Для разработки ПО, использующего различные алгоритмы планирования локальных траекторий колесных роботов и создания среды в симуляторе использовалась среда разработки Visual Studio Code (VSCode). В данной среде разработки соблюдаются все основные принципы диалога:

1. ***Пригодность для выполнения задачи***: для *VSCode* существует множество встроенных расширений, позволяющих проводить разработку, от-

ладку и сборку как отдельных модулей так и всего проекта. *VSCode* также поддерживает систему контроля версий *Git*, что ускоряет процесс разработки.

2. ***Информативность***: все заголовки и подзаголовки меню имеют понятное название и полностью предсказуемое поведение. Около каждого возможного действия из меню указаны горячие клавиши, вызывающее это действие. Все элементы связанные с организацией проекта имеют собственные иконки и всплывающие подсказки. Иерархия файлов проекта представлена в виде закрывающегося дерева, все популярные типы файлов имеют собственные иконки и могу быть дополнены расширениями. *VSCode* также отображает состояние всех исполняемых процессов в статусной панели.
3. ***Соответствие ожиданиям пользователя***: процесс сборки сопровождается оповещениями о текущем состоянии. Все ошибки замеченные во время написания кода отслеживаются и *VSCode* сразу указывает строки, файлы, и директории, которые содержат ошибки синтаксиса.
4. ***Пригодность для обучения***: *VSCode* содержит обширную документацию по всем компонентам среды, также предусмотрены руководства по использованию и разработке собственных расширений или аугментаций.
5. ***Управляемость***: *VSCode* имеет собственный интегрированный терминал, что позволяет сохранять историю всех запущенных процессов. Каждый процесс может быть запущен и остановлен в любое время. Все действия *VSCode* могут выполняться как мышью, так и горячими клавишами.
6. ***Устойчивость к ошибкам***: проект каждый промежуток времени кэшируется, что позволяет не беспокоиться за сохранность данных в случае отказа приложения. Удаление файлов требует подтверждения, а контроль версий позволит восстанавливать случайные изменения в проекте.

7. ***Пригодность для индивидуализации***: *VSCode* имеет полностью настраиваемую цветовую палитру. Все используемые шрифты могут быть изменены пользователем. Для каждого типа файлов может быть установлено пользовательское форматирование. Библиотека расширений *VSCode* имеет множество инструментов, позволяющих настроить среду для абсолютно любой задачи.

Следовательно, VSCode имеет эргономичный интерфейс и соответствует всем высокоуровневым принципам стандартов в ГОСТ Р ИСО 9241-100.

5.3 Оценка эргономики разработанной модели

Для оценки эргономики ПО, обеспечивающего проведение экспериментов необходимо определить из каких компонентов состоит система:

- Симулятор (Gazebo)
- Бортовое ПО (ROS)
- Визуализатор данных (Rviz)

Далее разобран каждый из компонентов.

Симулятор:

Предоставляет физически корректную среду, состоящую как из робота, так и окружающих его объектов. Состояние каждого объекта можно просмотреть в меню навигации. Модель робота и окружающие его препятствия могут иметь любую форму и характеристики, задаваемые пользователем. Время в симуляторе идет параллельно, но не всегда совпадает с реальным, например в моменты высокой нагрузки на систему. Сам симулятор широко поддерживается сообществом имеет множество плагинов и руководств. Так же симулятор поставляется вместе с контроллером, эмулирующим работу реального робота.

Бортовое ПО:

ROS не имеет графического интерфейса поскольку предназначен для исполнения на борту робота. Однако предоставлен широкий набор инструментов

для управления из командной строки. Архитектура ROS – это набор узлов, общающихся между собой, предоставляя тем самым гибкое управление устройством. Важным элементом является низкоуровневая реализация управления контроллерами робота, что решает основную задачу робототехники: связь ПО и аппаратной части робота. При отказе какого либо узла ROS, он по умолчанию перезапускается, что обеспечивает отказоустойчивость. Текущее состояние узлов, можно посмотреть с помощью *rqt_graph*, пример конфигурации, используемой в данной работе представлен на рисунке 1.

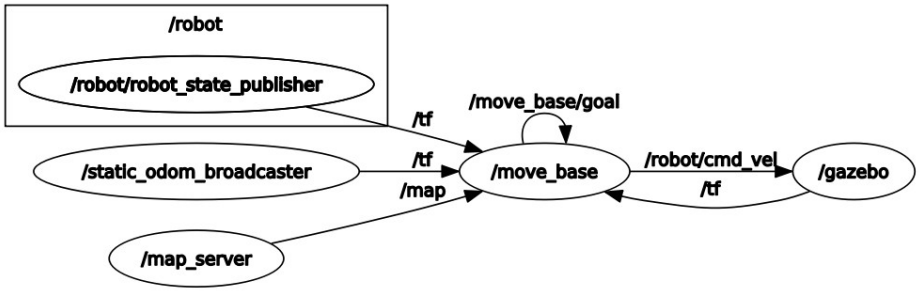

Рисунок 1. Пример конфигурации ROS (*rqt_graph*) для наземной навигации.

ROS также предоставляет инструментарий для регулировки узлов в реальном времени – *rqt_reconfigure*, пример настроек для локального планировщика показан на рисунке 2.

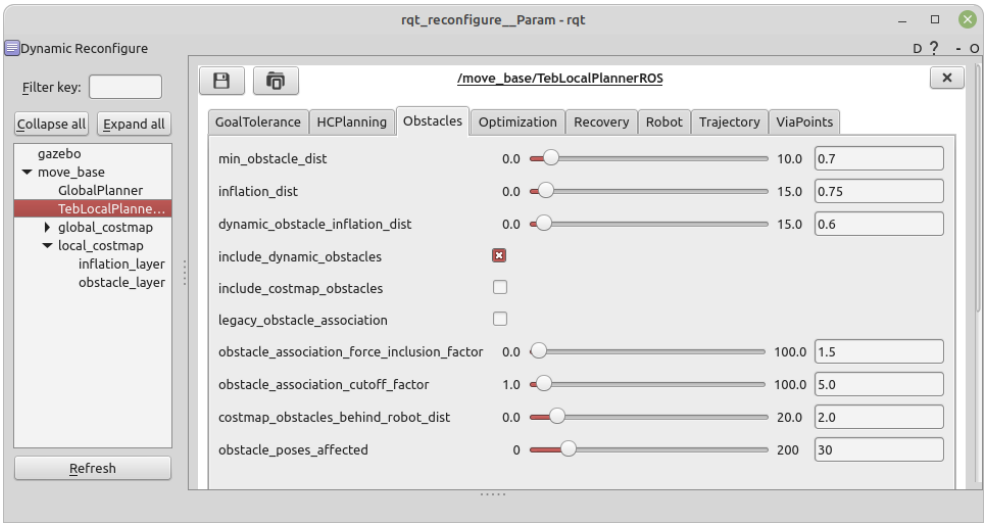

Рисунок 2. Окно настроек *rqt_reconfigure*.

Визуализатор: инструмент предназначенный для графической интерпретации данных, которыми оперируют узлы ROS. Данные можно выбрать как по типу сообщения так и с помощью топика, в который эти сообщения публикуются. Различные данные отображаются по разному: траектория в виде линии, позиция в виде стрелки с началом в указанной точке, окружение в виде тепловой карты и т. д. Если вдруг из каких-то топиков не отвечает, то его данные не отображаются, не нарушая при этом работу визуализатора. Пример визуализации представлен на рисунке 3.

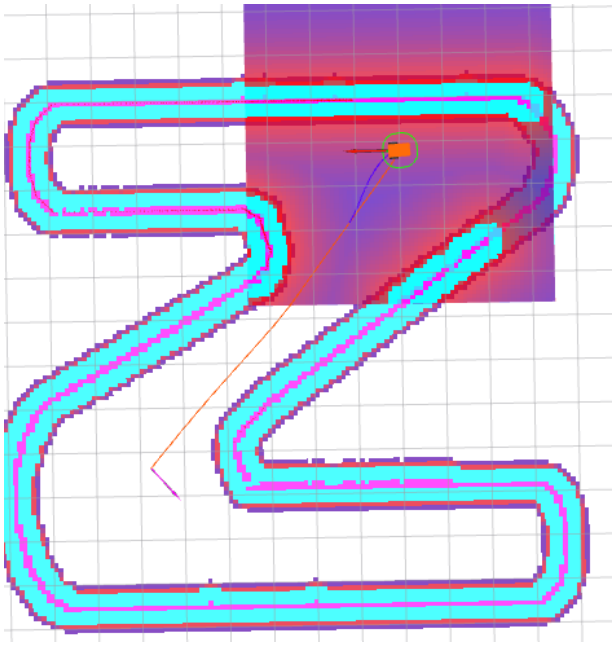

Рисунок 3. Пример визуализации данных в Rviz.

Разработанное ПО для проведения экспериментов по оценке алгоритмов локального планирования построено на базе рассмотренного ПО, следовательно удовлетворяет принципам диалога. Запуск каждого сервиса сопровождается сообщениями, описывающими текущий статус запуска, а затем статусом работы. Запущенный комплекс программ, ожидает публикации сообщения в топик «*goal*» ROS средствами командной строки или инструментом Rviz. Инструкция по запуску в приложении А.

ПРИЛОЖЕНИЕ А

Инструкция по запуску

Установка ROS

```
1.  sudo sh -c 'echo "deb http://packages.ros.org/ros/ubuntu
$(lsb_release -sc) main" > /etc/apt/sources.list.d/ros-lat-
est.list'
2.  sudo apt install curl # if you haven't already installed
curl
3.  curl -s https://raw.githubusercontent.com/ros/rosdistro/
master/ros.asc | sudo apt-key add -
4.  sudo apt install ros-noetic-desktop-full
5.  echo "source /opt/ros/noetic/setup.bash" >> ~/.bashrc
6.  source ~/.bashrc
7.  sudo apt install python3-rosdep python3-rosinstall
python3-rosinstall-generator python3-wstool build-essential
8.  sudo apt install python3-rosdep
9.  sudo rosdep init
10. rosdep update
```

Установка Gazebo

```
11. sudo sh -c 'echo "deb http://packages.osrfoundation.org/
gazebo/ubuntu-stable `lsb_release -cs` main" > /etc/apt/
sources.list.d/gazebo-stable.list'
12. wget https://packages.osrfoundation.org/gazebo.key -O - |
sudo apt-key add -
13. sudo apt-get update
14. sudo apt-get install gazebo11
```

Запуск симулятора

```
15. :local_planner_research $ cd simulation_ws
16. :local_planner_research $ catkin_make
17. :local_planner_research/simulation_ws $ source devel/set-
up.sh
18. # map can be <empty, zig-zag, corner, outdoor>
19. :local_planner_research/simulation_ws $ roslaunch
robot_gazebo robot_gazebo map:=empty
```

Запуск системы навигации

```
20. # after simulation run
21. :local_planner_research $ cd catkin_ws
22. :local_planner_research/catkin_make $ catkin_make
23. :local_planner_research/catkin_make $ source devel/set-
up.sh
24.
25. # availble arguments (can be undefined): map, local_plan-
ner, autogoal
26. # map -- map which used to build global plan. Available
<zig-zag, corner, outdoor>
27. # local_planner -- approach to build motion trajectory.
Available <dwa, tr, teb, mpc>
28. # auto_goal -- flag for automatically set goal pose and
start recording. Available <false, true>
29. :local_planner_research/catkin_make $ roslaunch nav_2d
robot_navigation
```