Оглавление

[1 Введение 4](#_Toc387509325)

[2 Программные и аппаратные средства разработки. 9](#_Toc387509326)

[2.1 ROS(Robotics Operation System) 9](#_Toc387509327)

[2.1.1 Компоненты ROS 9](#_Toc387509328)

[2.1.2 Файловая система ROS 12](#_Toc387509329)

[2.1.3 Вычислительный граф ROS 14](#_Toc387509330)

[2.2 iRobot Create. 16](#_Toc387509331)

[2.2.1 Подключаемые устройства 17](#_Toc387509332)

[2.2.2 Irobot Create+Kinect 17](#_Toc387509333)

[3 Алгоритмы построения пути 19](#_Toc387509334)

[3.1 Выбор алгоритма кратчайшего прохода по точкам 19](#_Toc387509335)

[3.2 Алгоритм объезда препятствий 20](#_Toc387509336)

[3.3 Движение робота 21](#_Toc387509337)

[4 Программная реализация 24](#_Toc387509338)

[4.1 Настройка среды для симуляции робота 24](#_Toc387509339)

[4.2 Модуль картографирования 25](#_Toc387509340)

[4.3 Запуск и тестирование алгоритма 26](#_Toc387509341)

[Список литературы 28](#_Toc387509342)

# 1 Введение

В недалёком прошлом, около лет 50 назад, роботы были чем-то сверхъестественным, и заметки о них можно было прочитать только на страницах фантастических книг и журналов, то теперь роботы не новость, и встречаются повсеместно. К примеру, стиральная машинка-автомат – тот же робот, запрограммированный на стирку наших вещей, или плеер – тоже робот. [Компьютер](http://your-happy-life.com/kompyuter-i-myi-nemnogo-o-windows/)– робот! Роботы ассоциируются с высокими технологиями будущего. Их применяют в самых различных сферах: для развлечений, освоения космоса, для медицины и конечно для нашей безопасности. Действительно, всё больше и больше новых технологий придумывают люди, чтобы облегчить себе жизнь, и называют их роботами. Но что это такое роботы и как они влияют на нашу жизнь? Не будем вдаваться в технические детали, но отметим, что робот – это автоматическое устройство, созданное по принципу живого организма, запрограммированное на то или иное действие. Причём, действие робот может совершать, как получая команды от человека, так и не получая (автономные и неавтономные роботы).

Робот в наше время очень полезная штука, именно благодаря им, человек облегчает себе жизнь. Возьмём простой пример – вычислительная робототехника. Одна из крупнейших американских корпораций уже занялась заменой менеджеров на робототехнику, в результате которой было уволено 4000 сотрудников! Менеджеров, и, кстати, очень эффективных, сокращают не из желания сэкономить, а лишь потому, что они просто не справляются с объёмом работы в отличие от роботов. Роботы не знают усталости и могут работать хоть круглые сутки, главное, чтобы хватило заряда батареек. А про эффективность роботов не следует и говорить – то, что у людей может занять год работы, робот выполнит за месяц. При этом эффективность работы роботов гораздо выше, чем у людей, и экономии больше. Роботам не надо платить, они не болеют, не могут получить травму, с работой справляются эффективней людей, а риск, что робот пострадает – гораздо меньше. Роботы обладают возможностями гораздо выше человеческих способностей, поэтому так часто нами применяются [1].

Робототехника является одной из самых быстроразвивающихся отраслей. Сейчас уже никого не удивишь роботами, способными выполнять несколько операций. Следующий шаг: многофункциональные роботы, роботы с искусственным интеллектом, способные сами принимать решения [2].

Например, робот PR2 уже способен на многие действия для помощи в быту человека.



Рисунок . PR2 жарит блинчики на кухне лаборатории Института искусственного интеллекта в Бремене, Германия.



Рисунок . PR2 доставляет почту.

Представьте себе персонального робота, который приносит кофе в постель; домашнего робота, помогающего пожилой бабушке, которая живет одна, по хозяйству; робоавтомобиль, который сам завозит в школу детей, а вас – на работу, пока вы спокойно завтракаете на заднем сидении...

Специалист по робототехнике из Университета Брауна Чед Дженкинс ожидает в скором времени революцию роботов, которая отзовется эхом компьютерной революции последних десятилетий. И он говорит, что это произойдет благодаря тому, что роботы смогут учиться, как люди: наблюдать, как ведут себя другие, задавать вопросы и применять свои знания в типичных ситуациях. «И эти технологии становятся все более дешевыми, более способными и более компактными», – добавляет он. В то время как сегодняшние роботы вооружены датчиками, сканерами, камерами и другими высокотехнологичными инструментами, специалисты все еще учатся помогать им понимать то, с чем они сталкиваются [3].

Менее функциональные роботы, такие как icreate, roomba нашли свое применение в быту как роботы пылесосы. Они являются необходимыми и самыми популярными бытовыми роботами в мире. В настоящий момент уже несколько миллионов роботов Roomba iRobot во всем мире занимаются уборкой домов с момента их дебюта в 2002 году. Однако и они могут быть использованы в простейших исследовательских целях – например, решение задачи пройти по маршруту [4].



Рисунок . iRobot Roomba.

Также на основе роботов Roomba или Create собрана модель turtlebot, которая помимо двигающейся основы включает в себя Microsoft Kinect и другие модули.

В данной работе решается задача движения iRobot Create (а также роботов, построенных на этой модели) по некоторой местности с возможностью задания координат и скоростей в заданных точках. Реализация данной задачи будет проходить в два этапа:

1. Моделирование поведения робота в программном симуляторе.
2. Перенос программы на реального робота.

Задача является актуальной в свете вышеозвученных проблем, возможность задания конкретных скоростей в точках маршрута может понадобиться в конвейерном производстве, в быту и других областях.

# 2 Программные и аппаратные средства разработки.

## 2.1 ROS(Robotics Operation System)

ROS была разработана Willow Garage – консорциумом Американских университетов в 2007 году. Целью ее создания была разработка мета-интерфейса программного обеспечения для роботов. В ROS использовался открытый исходный код, и первоначально она позаимствовала приложения из Player Project и других open source проектов. Она работала стабильно, проще устанавливалась, к ней имелась относительно хорошая документация, легкий для понимания интерфейс и она вела себя больше как операционная система. За пять лет своего существования, ROS имела семь релизов и за это время включила в себя огромный функционал: драйверы, алгоритмы, графические интерфейсы моделирования и парадигмы управления. ROS стала полезным инструментом для многих исследователей и ученых, которые поделились своим кодом с ROS сообществом.

### 2.1.1 Компоненты ROS

Рассмотрим каждый отдельный компонент ROS:

* Инфраструктура: ROS обеспечивает инфраструктуру коммуникации по схеме публикации-подписки сообщений (publish–subscribe messaging), предназначенную для быстрого и легкого построения распределенных вычислительных систем.
* Инструменты: ROS предоставляет широкий набор инструментов для настройки, запуска, самоанализа, отладки, визуализации, ведения списка событий, тестирования и остановки распределенной вычислительной системы.
* Возможности: ROS предоставляет широкий набор библиотек, реализующих полезные функции робота, с упором на мобильность, взаимодействие с предметами и восприятие окружающей среды.

Сообщество: ROS поддерживается и совершенствуется многочисленным сообществом, при этом очень большое внимание уделяется интеграции и документации.

Одна из самых поразительных особенностей ROS заключается в том, что она работает больше как операционная система, нежели как приложение. Во всех интерфейсах моделирования и управления используется архитектура клиент-сервер;

В ROS используются узлы (nodes), которые подсоединяются к мастеру (roscore), и являются соответственно клиентами и сервером. Существует два очень популярных симулятора. 2D симулятор – Stage и 3D симулятор – Gazebo.

Для Stage в ROS используется обертка – stageros, которая встраивает Stage в узлы ROS.

Gazebo позволяет моделировать в 3D, и поэтому использовать его не так просто, как Stage. Gazebo требователен к модели видеокарты и ресурсам системы.

Изначально Willow Garage установили ROS в разработанные ими роботы PR2 и Turtlebot. За последние годы было приложено много усилий, чтобы сделать ROS более аппаратно независимой и сейчас многие любительские, исследовательские и даже коммерческие роботы используют ROS для управления. Ученые, предприниматели, любители и промышленники часто представляют свой собственный пакет ROS для моделирования и управления конкретным роботом.

В Turtlebot и Billibot используется устройство Microsoft Kinect и, кроме того, в ROS частично поддерживается ей подобное устройство Xtion ASUS. В ROS также есть пакет для Android, так что роботом можно управлять, используя только мобильный телефон.

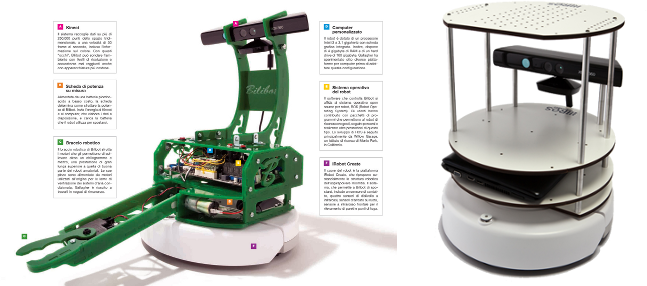


Рисунок . Модель робота Turtlebot.

Вот далеко не полный список роботов, использующих ROS: PR2, Turtlebot, Corobot, Roomba, KUKA youBot, Care-OBot, LEGO Mindstorms, AscTec Pelican/Hummingbird, Shadow Robot и Billibot [5].

ROS представляет собой надстройку над ОС, которая позволяет легко и просто разрабатывать системы управления роботами [6]. По сути, ROS – это набор из различных широко (и не очень) известных библиотек, таких как:

* OpenCV – библиотека, содержащая алгоритмы компьютерного зрения и обработки изображений c открытым исходным кодом. Она разрабатывается на С/С++, Python, Java и многих других языках;
* PCL (Point Cloud Library) – библиотека содержит множество алгоритмов для обработки 2D/3D изображений и облаков точек;
* OGRE (Object-Oriented Graphics Rendering Engine) – это обектно-ориентированный графический движок, написанный на С, с открытым исходным кодом. Он предназначен для упрощения и визуализации моделирования поведения робота;
* OROCOS (Open Robot Control Software) – библиотека для управления роботами, например, расчета кинематики;
* CARMEN (Carnegie Mellon Robot Navigation Toolkit) – библиотека для управления мобильными роботами. Она предназначена для выполнения базовых операций, необходимых мобильным роботам, таких как: сенсорное управление, обход препятствий, построение пути и создание карт.
* А также множество других полезных библиотек. Например, есть библиотека поддерживающая работу с Arduino. Помимо этого в ROS есть драйвера для различных манипуляторов, сенсоров (включая MS Kinect) и датчиков. Но что же отличает ROS от простой сборки библиотек? Основополагающим преимуществом является клиент-серверная архитектура ROS – разработчики реализовали механизм пересылки сообщений между различными объектами, возможность построения распределенных систем, ROS предоставляет интерфейсы для разработки программ на языках C++ и Python [7].

### 2.1.2 Файловая система ROS

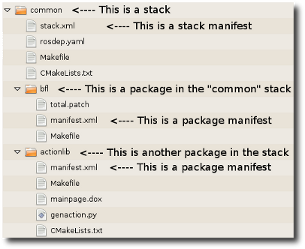
Понятие уровня файловой системы в основном охватывают ресурсы ROS, которые Вы встретите на диске:

* Пакеты: На уровне файловой системы основным блоком для организации программного обеспечения в ROS является пакет. Пакет ROS может содержать исходные коды и исполняемые файлы узлов, библиотеки, описание сообщений и сервисов, базы данных, файлы конфигурации, и другие ресурсы, которые логично организовать вместе. Каждый пакет должен содержать файл манифеста, который предоставляет метаданные о пакете, включая сведения о лицензии и зависимостях, а также флаги компилятора и так далее. Цель такого структурирования совершенно прозрачна – повышение возможности повторного использования [8].

Структура пакета выглядит следующим образом:

1. bin/: скомпилированные бинарники
2. include/package\_name: заголовочные файлы для C++ (обязательно должны описываться в manifest.xml!)
3. msg/: типы сообщений
4. src/package\_name/: исходный код на С++ и скрипты на Python'е, экспортируемые в другие пакеты
5. srv/: типы сервисов, предоставляемых пакетом
6. scripts/: скрипты на Python'е
7. CMakeLists.txt: CMake файл для сборки пакета
8. manifest.xml: манифест пакета
9. mainpage.dox: Doxygen-документация

В свою очередь, пакеты объединяются в стэки. На картинке вы можете увидеть пример такой структуры [9].



* Метапакет: представляет собой группу связанных пакетов.
* Пакет Манифеста: Манифесты (package.xml) объединяют метаданные о пакете, включая имя, версию, описание, информацию о лицензиях, зависимости и другие экспортируемые пакеты (предоставляют пакеты для других пакетов).
* Хранилища: содержат коллекцию пакетов.
* Сообщение (msg) типа: хранятся в my\_package / MSG / MyMessageType.msg
* Сервисы (srv): хранятся в my\_package / SRV / MyServiceType.srv, определяют запросы и структуры данных.

Все вышеперечисленное это статика ROS, о динамике в следующем разделе.

### 2.1.3 Вычислительный граф ROS

Уровень вычислительного графа – концепция представления данных в ROS в различных формах обработки. При работе ROS, строится «граф» – сеть точка-точка (peer-to-peer network) из процессов, которые связываются друг с другом через инфраструктуру ROS, пример на рис. 5.

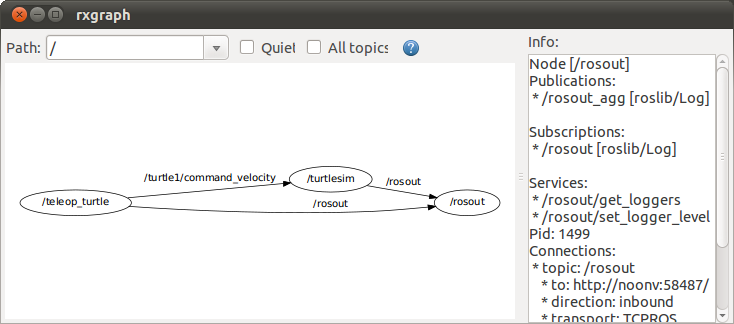


Рисунок . Пример графа ROS.

Основные составляющие вычислительного графа ROS:

* Узлы – это процессы, которые выполняют вычисления. ROS представляет из себя модульную систему, система управления роботом обычно включает в себя множество узлов, направленных на выполнение определённых функций. Каждый узел управляет каким-либо процессом, например, один узел управляет лазерным дальномером, один узел управляет колесными двигателями, один узел выполняет локализацию, один узел выполняет планирование пути, один узел обеспечивает графическое представление системы, и так далее. Узел ROS написан с использованием клиентских библиотек ROS, таких как roscpp или rospy.
* ROS Мастер обеспечивает связь между узлами. Без мастера узлы не смогут найти друг друга, обмениваться сообщениями, или вызывать сервисы.
* Сообщения – Узлы взаимодействуют друг с другом путем передачи сообщений. Сообщение – это структура данных, состоящая из полей с указанием типа поля. Поддерживаются стандартные типы (целые числа, с плавающей точкой, логические, и т.д.), а также массивы. Сообщения могут включать произвольно вложенные структуры и массивы (подобно синтаксису языка C).
* Темы – сообщения направляются через транспортную систему с семантикой писатель/подписчик. Узел посылает сообщение, опубликовав его в той или иной теме. Тема – это имя, которое используется для идентификации содержимого сообщения. Узел, который заинтересован в определенного рода данных может подключиться к нужной теме. Тема может иметь более одного писателя и более одного подписчика, один узел может писать и/или подписаться на несколько тем. В общем, писатели и подписчики не знают о существовании друг друга. Идея состоит в том, чтобы отделить производство информации от его потребления. Можно представить тему как строго типизированную шину сообщений.
* Сервисы: модель тем является очень гибким средством коммуникации, но его принцип многие-многим не подходит для взаимодействий вида запрос/ответ, которые часто требуются в распределенной системе. Такие запросы осуществляются через сервисы, которые определяются с помощью пары структур сообщений: одна для запроса и одна для ответа. Узел предлагает услугу под общим именем, клиент использует услугу путем отправки сообщения с запросом и ожидает ответ [10].

Вычислительный граф вместе с пакетами образует систему ROS.

## **2.2 iRobot Create.**

iRobot Create – [робот](http://607507.ru/wiostivlewmu2/%D0%A0%D0%BE%D0%B1%D0%BE%D1%82), разработанный компанией [iRobot](http://607507.ru/wiostivlewmu2/IRobot) на базе платформы робота-пылесоса [Roomba](http://607507.ru/wiostivlewmu2/Roomba). Представлен в [2007 году](http://607507.ru/wiostivlewmu2/2007_%D0%B3%D0%BE%D0%B4). Create предназначен для разработчиков роботов, позволяет программировать поведение робота. В отличие от [Roomba](http://607507.ru/wiostivlewmu2/Roomba), который так же (в моделях изготовленных после октября [2005 г](http://607507.ru/wiostivlewmu2/2005).) имел средства программирования поведения, Create разработан специально для данных задач и имеет более широкие возможности.

Одним из отличий от робота [Roomba](http://607507.ru/wiostivlewmu2/Roomba) является замена вакуумного оборудования (применяемого [Roomba](http://607507.ru/wiostivlewmu2/Roomba) для осуществления уборки) на специальный отсек для полезной нагрузки, снабженный 25-контактным разъёмом, используемым для связи с устанавливаемым в отсек оборудованием путём передачи цифровых или аналоговых сигналов. Create поддерживает [протокол](http://607507.ru/wiostivlewmu2/%D0%9F%D1%80%D0%BE%D1%82%D0%BE%D0%BA%D0%BE%D0%BB_%D0%BF%D0%B5%D1%80%D0%B5%D0%B4%D0%B0%D1%87%D0%B8_%D0%B4%D0%B0%D0%BD%D0%BD%D1%8B%D1%85) iRobot Roomba Open Interface.

### 2.2.1 Подключаемые устройства

Для Create поставляются практически все дополнительные [аксессуары](http://607507.ru/wiostivlewmu2/%D0%90%D0%BA%D1%81%D0%B5%D1%81%D1%81%D1%83%D0%B0%D1%80), разработанные компанией [iRobot](http://607507.ru/wiostivlewmu2/IRobot) для своих [домашних роботов](http://607507.ru/wiostivlewmu2/%D0%91%D1%8B%D1%82%D0%BE%D0%B2%D0%BE%D0%B9_%D1%80%D0%BE%D0%B1%D0%BE%D1%82). Так же дополнительно поставляется «командный модуль» включающий [микроконтроллер](http://607507.ru/wiostivlewmu2/%D0%9C%D0%B8%D0%BA%D1%80%D0%BE%D0%BA%D0%BE%D0%BD%D1%82%D1%80%D0%BE%D0%BB%D0%BB%D0%B5%D1%80) с интерфейсом [USB](http://607507.ru/wiostivlewmu2/USB) и четырьмя дополнительными разъёмами с интерфейсом [DE-9](http://607507.ru/wiostivlewmu2/D-subminiature).

Робот Create может управляться внешним компьютером, что позволяет преодолеть проблемы связанные с ограниченной вычислительной мощностью и ограниченным объёмом памяти встроенного вычислителя «командного модуля». Для этих целей часто используются миниатюрные компьютеры [Gumstix](http://607507.ru/) [11].

### 2.2.2 Irobot Create+Kinect

Это робот, который может создавать 3D-карту окружающего пространства. Irobot Create "носит" на себе платформу Kinect, способную записывать данные об окружающей местности и переносить их в виде 3D-изображения в ваш компьютер, с помощью сенсоров сканирует обстановку и передает полученные данные. И всё это есть в TurtleBot [12].



Рисунок . Turtlebot как расширение iRobot Create.

TurtleBot с первого взгляда напоминает белую многоярусную табуретку, места на ярусах зарезервированы для ноутбука и сенсора Kinect. Грубо говоря, это мобильная база для создания собственного мобильного робота. Но TurtleBot – это не только железо.

Это ещё и TurtleBot SDK – комплект разработчика, в который входят практически все необходимые инструменты для того, чтобы заставить робота делать то, что вы от него хотите (кстати, ПО, которое в него входит, опенсорсное и разработано в рамках сообщества ROS) [13].

# 3 Алгоритмы построения пути

Решение задачи построения пути для движения робота разделяется на две связанные друг с другом подзадачи:

* Выбор алгоритма кратчайшего прохода по точкам
* Объезд препятствий

Как известно, задача нахождения кратчайшего маршрута между N точками является классической задачей коммивояжера с добавлением препятствий. Влияние препятствий приводит к тому, что расстояние из каждой точки в каждую точку приходится рассчитывать с учетом объезда возможных препятствий между ними.

К примеру, на рис. 7 без препятствий кратчайший путь робота выглядел бы O1 – O2 – O3, однако очевидно, что с учетом препятствия, данный путь уже не будет кратчайшим, но кратчайшим является путь O1 – O3 – O2.



Рисунок . Пример обхода препятствия.

## 3.1 Выбор алгоритма кратчайшего прохода по точкам

Для решения классической задачи коммивояжера [14] существует множество приближенных методов, для построения маршрута для робота можно воспользоваться одними из наиболее простых – жадными методами, к примеру – метод ближайшего соседа. В задаче с препятствиями этот метод необходимо модернизировать – расстояние между 2 точками считается с учетом объезда препятствий.

Применение более сложных методов поиска пути даст выигрыш в редких случаях, однако будет гораздо больше нагружать вычислительные ресурсы робота.

## 3.2 Алгоритм построения пути между двумя точками

Навигация в ROS разделяется на три отдельных модуля:

* Глобальный планировщик маршрута
* Локальный планировщик маршрута
* Модуль выхода из тупиковых ситуаций

Глобальный планировщик создает общий план маршрута, разделив его на путевые точки а также передает этот план локальному планировщику.

Локальный планировщик это модуль, отвечающий за наиболее оптимальное перемещение робота между двумя путевыми точками, установленными глобальным планировщиком. Именно локальный планировщик содержит алгоритмы для прокладывания оптимального пути между препятствиями. В данной работе используется алгоритм динамического окна (DWA – dynamic window approach):

* Фазовое пространство управления движением робота разбивается дискретно (, , ).
* Для каждой отобранной скорости выполняется моделирование движения робота от текущего состояния, чтобы предсказать, что произойдет, если применять данные скорости в течение некоторого (короткого) периода времени.
* В результате моделирования каждая траектория оценивается, используя метрику, которая включает такие характеристики, как: близость к препятствиям, близость к цели, близость к глобальному пути и скорости. Отбрасываются траектории, ведущие к столкновению.
* Выбирается траектория с наибольшим значением метрики и значения скоростей отправляются роботу.
* Повторять алгоритм пока робот не достигнет точки цели.

## 3.3 Движение робота с указанием скоростей в точках

Навигация ROS предполагает движение точка-точка, при этом робот разгоняется и тормозит с ускорением a, его скорость не превышает .

Навигация ROS не предполагает задание скоростей в конечных точках маршрута, однако данное условие можно выполнить задавая точки косвенно. Например, чтобы иметь в точке О заданные скорость и направление, роботу нужно передать координаты точки О\*, находящейся дальше по пути за точкой О, необходимо только рассчитать координаты этой точки.



Рисунок . Фиктивная точка пути для обеспечения скорости в заданной точке.

Рассчитаем эту точку, пусть координаты и скорости точки О соответственно (), тогда из кинематики:

Тогда координаты фиктивной точки рассчитываются следующим образом:

Учитывая, что:

Получим окончательные формулы:

При достижении роботом заданной точки, текущее задание – двигаться в фиктивную точку отменяется и робот продолжает движение к следующей точке маршрута.

# 4 Программная реализация

Реализация алгоритма происходит в два этапа, первый включает в себя реализацию алгоритма и его тестирование в рамках программной симуляции, второй этап – запуск алгоритма на реальном роботе и его корректировка.

Для работы алгоритма как в симуляторе, так и на реальном роботе используется общая платформа ROS.

## 4.1 Настройка среды для симуляции робота

Для осуществления работы алгоритма на симуляции требуется установка операционной системы Ubuntu 12.04 и метапакета ROS, включая следующие пакеты:

* ros-hydro-turtlebot-\* – все, что связано с роботом turtlebot;
* ros-hydro-gmapping – модуль картографирования
* gazebo – программа для симуляции
* rviz – визуализация сообщений робота (например, построенная карта)

Запуск симуляции производится следующими действиями:

* Запуск робота в среде gazebo(можно писать собственный launch скрипт или использовать существующие)

roslaunch turtlebot\_gazebo turtlebot\_playground.launch

* Запуск модуля картографирования

roslaunch turtlebot\_gazebo gmapping\_demo.launch

* Запуск визуализации генерируемой карты

roslaunch turtlebot\_rviz\_launchers view\_navigation.launch

* Запуск управления роботом с клавиатуры

roslaunch turtlebot\_teleop keyboard\_teleop.launch

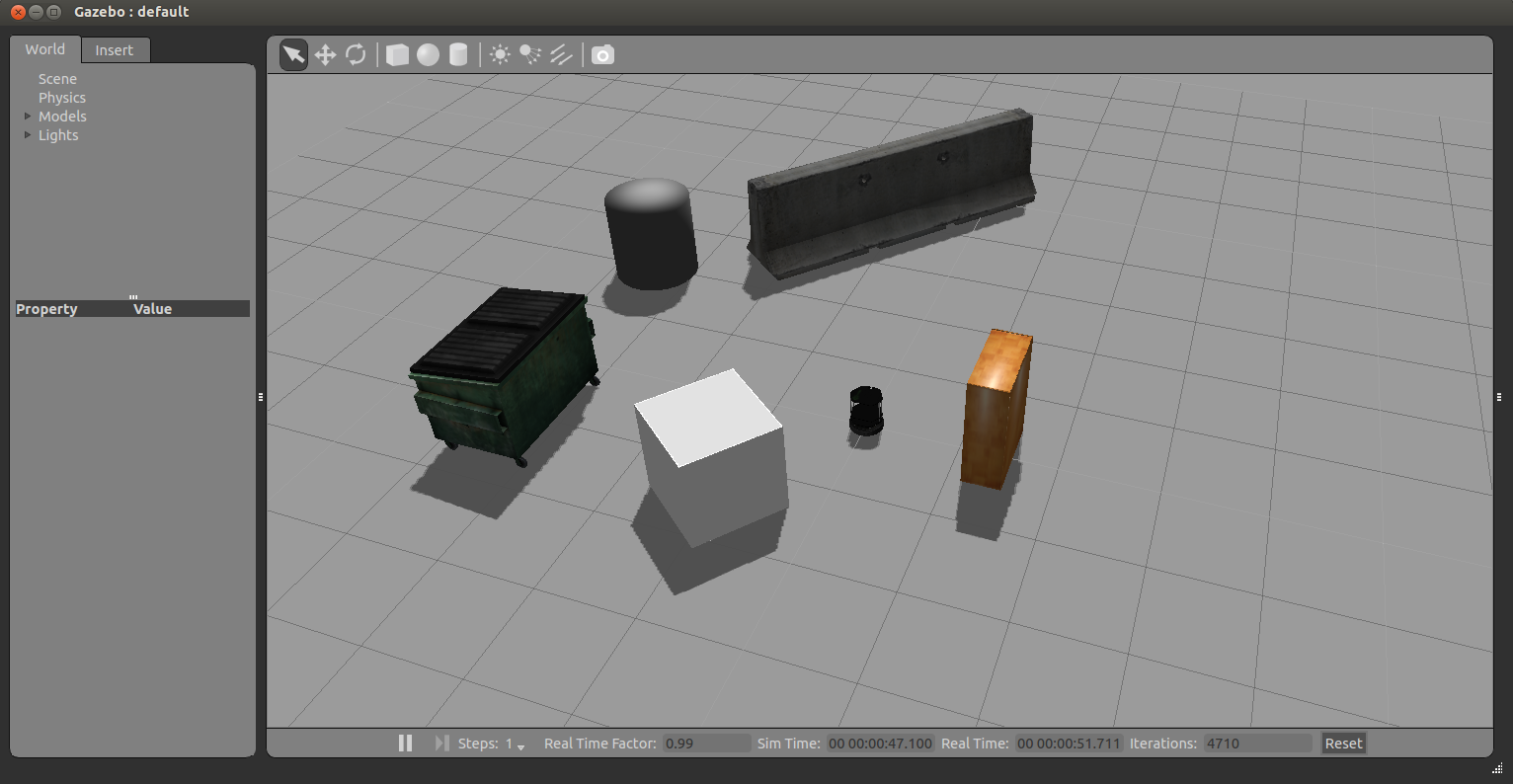


Рисунок . Симуляция робота в gazebo.

После запуска симуляции и картографирования возможно запускать реализации каких-либо алгоритмов для робота.

## 4.2 Модуль картографирования

В данной работе используется стандартный модуль картографирования gmapping, работающий по технологии SLAM (simultaneous localization and mapping, одновременная локализация и картографирование). Данная технология позволяет роботу, имеющему устройство лазерного сканирования, одновременно и составлять карту, и определять свое местоположение на карте. Пример карты показан на рис. 11.

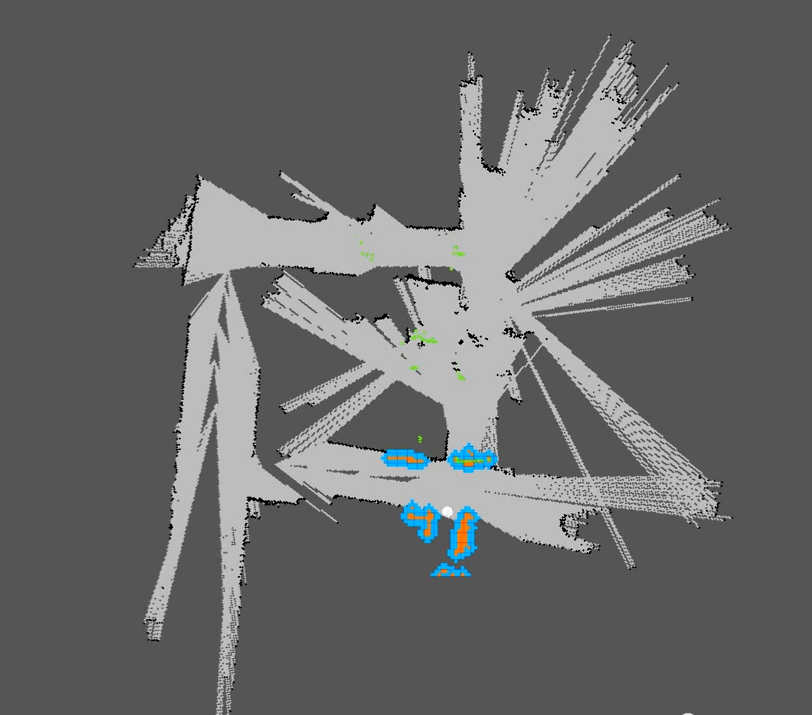


Рисунок . Пример работы модуля картографии.

Модуль картографирования может работать как с симуляцией, так и с реальным роботом.

## 4.3 Запуск и тестирование алгоритма

Для того, чтобы запустить реализованный алгоритм необходимо наличие программы-сборщика catkin и его рабочего пространства. Алгоритм собирается в пакет ROS и может быть запущен следующим образом:

rosrun turtlebot\_program turtlebot\_program

Алгоритм строит оптимальный маршрут для робота по главе 3, затем дает команды роботу для прохождения этого маршрута.

В алгоритме используется тема(topic) /map для получения карты и ориентации робота в пространстве, а также тема(topic) /cmd\_vel/input/teleop для передачи сообщений, включающих в себя линейную и угловую скорость робота. Робот обходит все заданные точки (по умолчанию все объекты в поле видимости) по описанным в данной работе алгоритмам.

# Список литературы

|  |  |  |
| --- | --- | --- |
| [1] | [В Интернете]. http://your-happy-life.com/robotyi-v-nashey-zhizni/. | |
| [2] | [В Интернете]. http://photochronograph.ru/2013/05/18/roboty-voshedshie-v-nashu-zhizn/. | |
| [3] | [В Интернете]. http://www.km.ru/science-tech/2013/07/29/robototekhnika/716802-issledovateli-nauchili-robota-obuchatsya-uvidennomu. | |
| [4] | [В Интернете]. http://myrobot.com.ua/about\_irobot/. | |
| [5] | [В Интернете]. http://test.wertech.ru/Blog/42-open-source-.aspx. | |
| [6] | [В Интернете]. http://habrahabr.ru/post/128024/. |
| [7] | [В Интернете]. http://robome.ru/ros\_ili\_zachem\_pisat\_odno\_i\_tozhe/. |
| [8] | [В Интернете]. http://robocraft.ru/blog/technology/451.html. |
| [9] | [В Интернете]. http://habrahabr.ru/post/128024/. |
| [10] | [В Интернете]. http://wiki.ros.org/ROS/Concepts. |
| [11] | [В Интернете]. http://ru.wikipedia.org/wiki/IRobot\_Create. |
| [12] | [В Интернете]. http://www.robotmart.ru/news/detail/78. |
| [13] | [В Интернете]. http://www.computerra.ru/11257/turtlebot-kinect-na-kolyosah/. |
| [14] | [В Интернете]. http://ru.wikipedia.org/wiki/Задача\_коммивояжера. |
| [15] | [В Интернете]. http://ru.wikipedia.org/wiki/Алгоритм\_Ли. |