Министерство науки и высшего образования Российской Федерации
Федеральное государственное автономное образовательное учреждение высшего образования
«КАЗАНСКИЙ (ПРИВОЛЖСКИЙ) ФЕДЕРАЛЬНЫЙ УНИВЕРСИТЕТ»
ИНСТИТУТ ИНФОРМАЦИОННЫХ ТЕХНОЛОГИЙ И ИНТЕЛЛЕКТУАЛЬНЫХ СИСТЕМ

Направление: 09.04.04 – Программная инженерия
Магистерская программа: Робототехника

ВЫПУСКНАЯ КВАЛИФИКАЦИОННАЯ РАБОТА

МОДЕЛИРОВАНИЕ РОБОТА СЕРВОСИЛА ИНЖЕНЕР И ПЛАНИРОВАНИЕ МАРШРУТА ЕГО ДВИЖЕНИЯ В 2.5D С УЧЕТОМ МАССОВО-ГАБАРИТНЫХ ПАРАМЕТРОВ

Обучающийся 2 курса
группы 11-931
Доброквашина А.С.

Научный руководитель
PhD (техничекие науки),
профессор кафедры
интеллектуальной робототехники
Магид Е.А.

Директор ИТИС КФУ
канд. техн. наук
Абрамский М.М.

.

Казань – 2021 г.

СОДЕРЖАНИЕ

ВВЕДЕНИЕ 5

ОБЗОР ЛИТЕРАТУРЫ 7

ГЛОССАРИЙ 11

ЧАСТЬ 1. ОБЗОР ИНСТРУМЕНТОВ 12

1.1. ROS (Robot Operating System) 12

1.2. Симулятор Gazebo 13

1.3. Визуализатор RViz 14

1.4. Симулятор Webots 14

ЧАСТЬ 2. РОБОТ "СЕРВОСИЛА ИНЖЕНЕР" 16

ЧАСТЬ 3. РАЗРАБОТКА СИМУЛЯЦИОННОЙ МОДЕЛИ РОБОТА "СЕРВОСИЛА ИНЖЕНЕР" 20

3.1. Исходная симуляционная модель робота “Сервосила Инженер” 20

3.2. Изменения в симуляционной модели робота "Сервосила Инженер" в симуляторе Gazebo 22

3.2.1. Модели коллизий 22

3.2.2. Визуальные модели 26

3.2.3. Конструкция модели робота 27

3.2.4. Симуляция гусениц 30

3.2.5. Сенсоры и камеры 32

3.2.6. Инерции 37

3.3. Создание симуляционной модели робота "Сервосила Инженер" в симуляторе Webots 41

ЧАСТЬ 4. СТЕК НАВИГАЦИИ 43

4.1. Добавление стека навигации к симуляционной модели робота "Сервосила Инженер" в симуляторе Gazebo 43

4.2. Эксперименты со стеком навигации 44

4.2.1. Реальный робот 44

4.2.2. Симуляционная модель в Gazebo 46

ЧАСТЬ 5. ДВИЖЕНИЕ ПО НЕРОВНЫМ ПОВЕРХНОСТЯМ 49

5.1. Создание симуляционной модели RSE в симуляторе Gazebo 49

5.2. Эксперименты с движениями по неровным поверхностям 51

5.2.1. Симуляционная модель в Gazebo 51

5.2.2. Симуляционная модель в Webots 53

5.2.3. Реальный робот 54

ЗАКЛЮЧЕНИЕ 55

СПИСОК ЛИТЕРАТУРЫ 57

ПРИЛОЖЕНИЯ 62

Приложение А 62

А1. Фрагмент UDRF файла с описанием манипулятора робота “Сервосила Инженер” - engineer_4dof_arm.xacro 62

А2. Фрагмент UDRF файла с описанием симуляционных колес робота “Сервосила Инженер” - diff_drive.xacro 70

А3. Фрагмент UDRF файла с захвата робота “Сервосила Инженер” - engineer_ee.xacro 72

А4. Фрагмент UDRF файла сописанием базы робота “Сервосила Инженер” - engineer_mobile_base.xacro 76

Приложение Б 77

Фрагмент UDRF файла с описанием симуляционной модели Random Step Environment - RSE.xacro 77

Приложение В 81

В1. Параметры локального планировщика - local_planner_params.yaml 81

В2.Параметры локальной карты стоимости - local_costmap_params.yaml 82

В3. Параметры глобальной карты стоимости - global_costmap_params.yaml 83

ВВЕДЕНИЕ

Главной целью робототехники была и всегда будет помощь людям. В настоящее время роботов можно встретить практически в любой сфере человеческой жизни. Они собирают машины, убирают дома, доставляют посылки и помогают спасать людские жизни за операционным столом.

Поисково-спасательная робототехника (Urban Search and Rescue, USAR) — одно из популярных направлений применения роботов. Существует огромное количество способов применения подобной техники в таких экстремальных условиях, как природные и техногенные катастрофы, опасные производства и т.д. Так, например, первые испытания роботов-спасателей в реальных условиях были проведены во время событий во Всемирном торговом центре в 2001 году. Центр поиска и спасения с помощью роботов (CRASAR) восемь раз использовал радиоуправляемых роботов разных размеров [34]. Важность USAR-робототехники также подчеркнула ситуация, произошедшая в Фукусиме [35]. Работа в условиях радиоактивного загрязнения, смертельно опасного для человека, не представляется проблемой для специализированного робота. Более того, большое количество различных конфигураций роботов, может сделать их не просто хорошей заменой человеку в определенных задачах, но и предложить более быстрое и качественное выполнение некоторых видов работ.

Одними из наиболее полезных типов роботов в условиях неровных поверхностей и труднопроходимых областей, которыми зачастую характеризуются ситуации, возникающие в поисково-спасательных операциях, являются гусеничные роботы. [22, 23] Благодаря гусеницам проходимость таких роботов может быть чрезвычайно высокой [14, 15, 18]. По этой причине они широко используются в различных исследованиях и проектах [11, 12 16, 21]. В данной работе речь пойдет об одном из подобных роботов, гусеничном роботе российского производства “Сервосила Инженер”.

При работе с поисково-спасательной робототехникой среда для тестирования созданных и уже существующих алгоритмов может быть крайне сложной. Такие обстоятельства, как обломки зданий и загрязнение воздуха, не легко воспроизвести в повседневной жизни, и именно для этих целей были созданы робототехнические симуляторы. Они предоставляют возможность создавать виртуальные среды для проведения различных экспериментов с роботом. По этой причине, почти все роботы имеют собственные симуляционные модели.

Ранее уже производились попытки создать симуляционную модель для робота “Сервосила Инженер” [1, 2]. Благодаря ним уже имеются некоторые наработки в сфере симуляции гусениц.

Целью данной работы является создание полноценной модели для данного робота, с соблюдением массово-габаритных параметров, воссозданием всей визуальной составляющей, а также с добавлением сенсоров, которыми оборудован реальный робот.

Среди задач, поставленных в работе: создание симуляционной модели робота «Сервосила Инженер», добавление стека навигации для симуляционной модели робота и его настройка как для симуляционной модели, так и на реальном роботе, а также проведение серии экспериментов с навигацией и движением робота в условиях неровных поверхностей.

ОБЗОР ЛИТЕРАТУРЫ

На сегодняшний день существует ряд исследовательских работ посвященных созданию симуляционных моделей роботов. В условиях поставленной задачи большой интерес представляют работы по симуляции гусениц и гусеничных роботов. Также были рассмотрены несколько статей, посвященных итогам предыдущих исследований на тему создания симуляционной модели робота «Сервосила Инженер».

Так, например, работа [1] рассказывает о моделировании и симуляции робота “Сервосила Инженер” с помощью ROS/Gazebo. Она содержит в себе основную информацию о роботе и его возможностях, а также о симуляции. Данная статья является основополагающей в теме, касающейся симуляционной модели. В статье используется Gazebo версии 2.2.3 и ROS Indigo, тогда как сейчас эти версии симулятора и операционной системы уже не поддерживаются. Это одна из первых работ, связанных с роботом «Сервосила Инженер», в которой упоминаются сложности, касающиеся распределения веса при работе с манипулятором.

Работа [2] продолжает тему моделировании и симуляции робота “Сервосила Инженер” с помощью комплекса ROS/Gazebo, и более глубоко затрагивает тему симуляции гусениц с помощью инструмента Gazebo-tracks. Предложенный инструмент является интересным решением, однако ROS-пакет, содержащий его, был разработан в 2013 году, и с тех пор никаких комментариев и изменений в его код не вносилось. Все это дает основания полагать, что пакет Gazebo-tracks не поддерживается. В таком случае его поведение в новых версиях ROS и Gazebo может быть крайне нестабильным.

Следующая работа [3] также затрагивает тему симуляции гусениц и предлагает более универсальное решение. Данное исследование рассматривает метод симуляции гусениц с помощью массива небольших колес. В статье рассмотрены несколько вариантов симуляций с использованием разного

количества колес. Все варианты были сравнены между собой и был найден самый оптимальный из них.

Далее будут рассмотрены несколько исследований, касающихся симуляции гусениц. Эта тема является достаточно популярной среди робототехников, работающих с моделями гусеничных роботов в симуляторе Gazebo. Причиной этому стал тот факт, что данный симулятор не имеет встроенных инструментов для симуляции гусениц.

В работе [4] рассматриваются четыре способа симуляции: деформируемые гусеницы, гусеницы, представленные несколькими колесами, гусеницы, выполненные с помощью пластин и не деформируемые гусеницы. Исследование предлагает детальное сравнение точности и сложности подсчетов для всех предложенных способов, а также практическое сравнение в условиях симуляции. Благодаря тестам, проведенным авторами, можно четко определить плюсы и минусы этих представлений. Так, например, выполнение гусениц в виде взаимосвязанных пластин дает максимально точное повторение их поведения в реальности, однако является сложнейшим решением в плане подсчетов для компьютера. Это делает решение крайне ресурсозатратным, что приведет к резкому снижению фактора реального времени (RTF) и сделает симуляцию крайне медленной.

Еще одно исследование в области работы с гусеничными роботами представлено в [5]. Данная статья рассматривает возможности передвижения и преодоления препятствий роботом SHU-I с шестью гусеницами, четыре из которых имеют возможность автономного движения. Конструкция робота SHU-I схожа с роботом “Сервосила Инженер” за исключением того факта, что последний оборудован лишь двумя активными флипперами. В статье представлена детализированная и хорошо проиллюстрированная симуляционная модель гусеничного робота, которая послужила хорошим примером при создании симуляционной модели робота “Сервосила Инженер”.

Следующие работы, которые будут рассмотрены, в большей степени помогают ознакомиться с самим роботом. Так, например, исследование [6] рассказывает о внедрении в конструкцию робота “Сервосила Инженер” дополнительного лазерного сенсора — лидара. Также описывается способ его подключения к самому роботу, и предоставляется информация о некоторых экспериментах, связанных с работой добавленного сенсора. В частности, упоминается опыт использования алгоритма LRF-SLAM для одновременного картографирования и определения местоположения робота.

Для ознакомления с самим роботом были рассмотрены исследования, посвященные управлению роботом и разработке графического и мобильного интерфейсов для управление роботом. Работы [7, 8, 25] описывают способ связи с роботом “Сервосила Инженер”, а также типы пакетов и команд, передаваемых роботу. Вся эта информация создает четкое представление, как о границах подвижности частей робота, так и в целом о его работе.

Робот “Сервосила Инженер” оснащен двумя активными флипперами, в связи с чем было решено также рассмотреть исследования, целью которых стало использование активных флипперов для преодоления различных препятствий. Работа [9] рассказывает о создании алгоритма для автономной работы гусеничных флипперов при преодолении препятствий. В работе присутствует довольно детальное, хорошо проиллюстрированное описание алгоритма работы флипперов. Однако стоит заметить, что у робота, описанного в статье, имеется четыре независимых флиппера, тогда как у робота “Сервосила Инженер” их только два и двигаться они могут лишь синхронно.

Одной из тем, затронутых при обзоре литературы, стало передвижение по неровным поверхностям. Для этого была изучена серия работ, связанных с алгоритмами движения в условиях неровных поверхностей [10, 13, 17], а именно сохранение и предумышленная потеря равновесия. Помимо четко описанных алгоритмов, была также отмечена конструкция RSE (Random Step Environment), которая использовалась в экспериментах. Данная конструкция нашла свое

применение в рамках работы при проведении экспериментов как с реальным роботом, так и с его симуляционной моделью.

Были рассмотрены как работы, связанные напрямую с роботом “Сервосила Инженер”, так и вспомогательные работы, которые смогут помочь в некоторых конкретных задачах моделирования, а также смогут облегчить проведение экспериментов.

ГЛОССАРИЙ

RTF (Real Time Factor) — коэффициент соотношения реального времени с симуляционным; так, например, RTF равный 0,5 будет означать, что для воспроизведения минутного эксперимента в реальном времени, симулятору понадобиться две минуты, и каждая секунда симуляции будет просчитываться около двух секунд реального времени.

ROS-нода — это специальный исполняемый файл, который взаимодействует с другими нодами посредством ROS-топиков и ROS-сервисов.

ROS-топик — это поток данных одной структуры. Любая нода может публиковать данные в топик или получать их из него. Является одним из способов общения ROS-нод между собой.

Лидар (Лазерный дальномер) — устройство, измеряющее расстояние до окружающих объектов, посредством испускания света и замера скорости его отражения.

IMU (Inertial Measurement Unit) датчик — сенсор, представляющий собой гиростабилизатор. Он дает информацию о скорости и ускорении движения робота, а также о его ориентации в пространстве.

URDF (Unified Robot Description Format) — унифицированный формат описания робота. URDF файл содержит в себе описание робота, состоящее из описания всех составляющих этого робота.

ЧАСТЬ 1. ОБЗОР ИНСТРУМЕНТОВ

1.1. ROS (Robot Operating System)

Robot Operating System (Робототехническая операционная система, ROS) это комплекс подходов и инструментов для программирования робототехнических систем. Она предоставляет возможность работать со всеми аспектами системы, начиная с управления моторами одного робота и заканчивая взаимодействием роботов между собой.

ROS предоставляет пользователям широкий выбор различных готовых решений и библиотек. Система поддерживает большое количество языков программирования, среди которых Java, Go, Lua, Javascript, C#, однако самыми популярными и широко поддерживаемыми на сегодняшний день являются C++ и Python.

Другой особенностью ROS является тот факт, что система полностью обеспечивает связь между отдельными программными компонентами (нодами) и пакетами посредством топиков и сервисов. Такая универсальность позволяет свободно взаимодействовать нодам и пакетам, написанным на разных языках программирования. Эта особенность позволяет в пределах одного проекта, при необходимости, использовать несколько языков программирования, используя каждый из них для специализированных задач.

Робототехническая операционная система имеет открытый исходный код и массу различных библиотек, доступных для использования как в учебных, так и в коммерческих целях. Доступность ROS обеспечивает данную операционную систему большим комьюнити, что помогает быстро решать многие из возникающих проблем. Все вышеописанное делает ROS крайне популярным и востребованным инструментом при программировании роботов.

1.2. Симулятор Gazebo

Gazebo является одним из популярных робототехнических симуляторов благодаря своей совместимости с ROS. Данный симулятор устанавливается вместе с робототехнической операционной системой, а большинство роботов, работающих с ROS, имеют симуляционные модели, построенные именно в формате, доступном симулятору Gazebo.

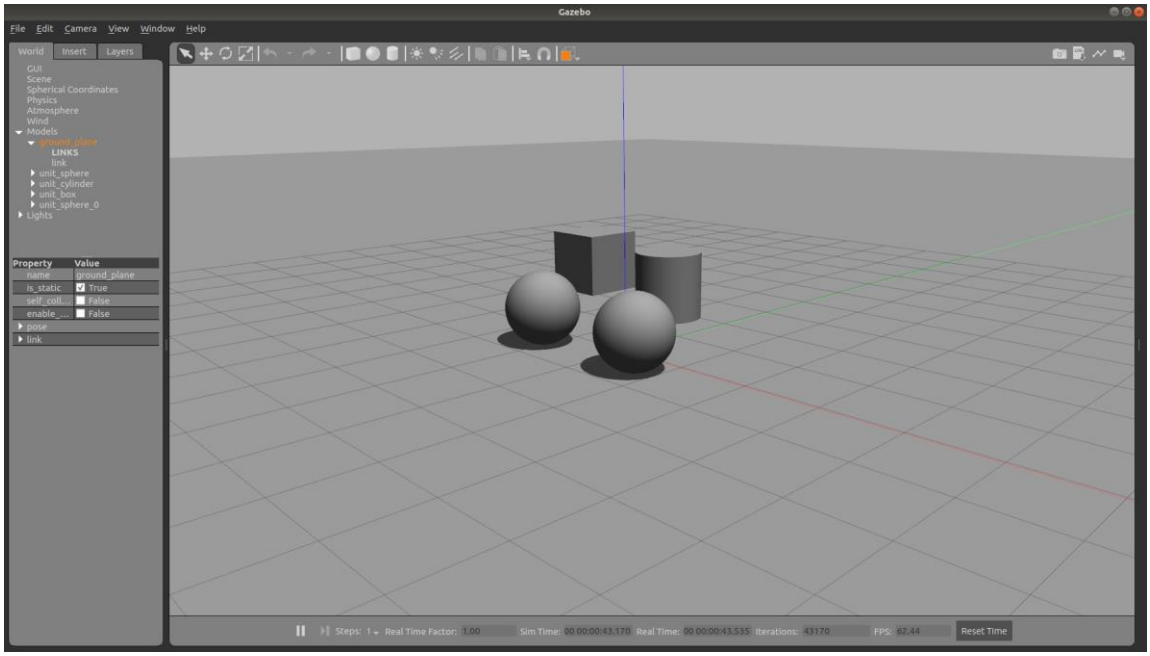

Рисунок 1.2.1. Интерфейс симулятора Gazebo.

Сам симулятор работает с использованием физического движка Open Dynamics Engine (ODE), а для визуализации использует OpenGL. Gazebo имеет достаточно достоверную симуляцию физики и различных физических явлений, а также имеет большое количество плагинов для симуляции работы датчиков, например лидаров или камер. На рисунке 1.2.1 представлен пример интерфейса симулятора.

1.3. Визуализатор RViz

RViz представляет собой инструмент визуализации, используемый в ROS. В числе его возможностей отображение модели робота, данных с различных сенсоров и датчиков, управление контроллерами, контроль работы стека навигации и прочее.

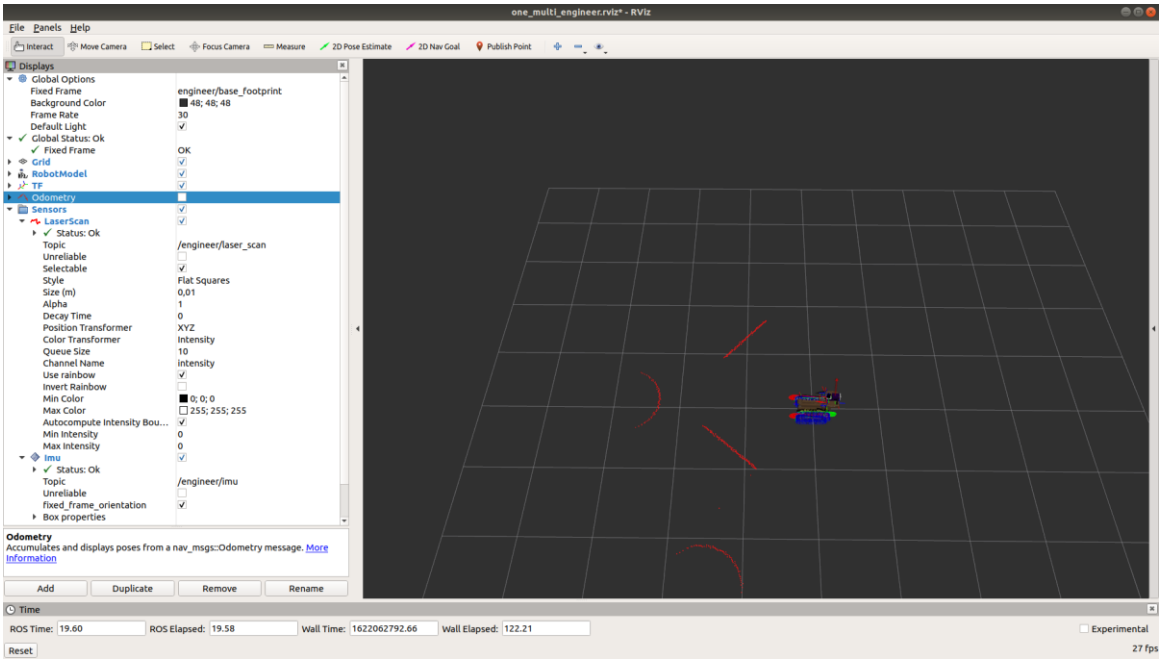

Рисунок 1.3.1. Интерфейс визуализатора RViz.

На рисунке 1.3.1 изображен пример интерфейса RViz с отображением модели робота, а также данных с его сенсоров — лидара и IMU.

1.4. Симулятор Webots

Webots — робототехнический симулятор компании Cyberbotics. Широко используется для симуляции различных робототехнических проектов [26]. В отличии от Gazebo поддержка работы с ROS является дополнительной функцией в Webots.

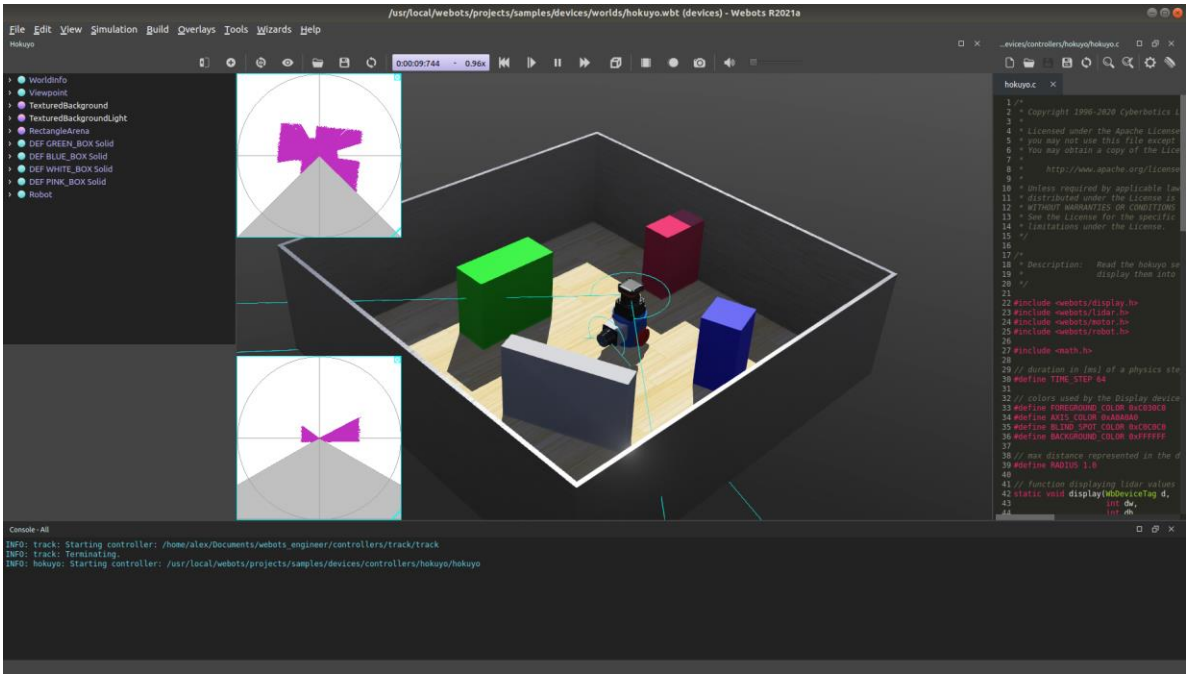

Рисунок 1.4.1. Интерфейс симулятора Webots.

Данный симулятор является полностью самостоятельным продуктом. Он поддерживает создание новых роботов с нуля с помощью визуального конструктора, а также написание контроллеров и скриптов непосредственно в самом симуляторе. Так на рисунке 1.4.1 можно увидеть пример графического интерфейса симулятора Webots. В данном случае открыт один из шаблонных проектов, в котором используется датчик — лидар. В симуляторе отображаются графически представленные данные с лидара, скрипт, отвечающий за движение датчика, а также сама симуляционная среда, в которой происходит это движение.

ЧАСТЬ 2. РОБОТ "СЕРВОСИЛА ИНЖЕНЕР"

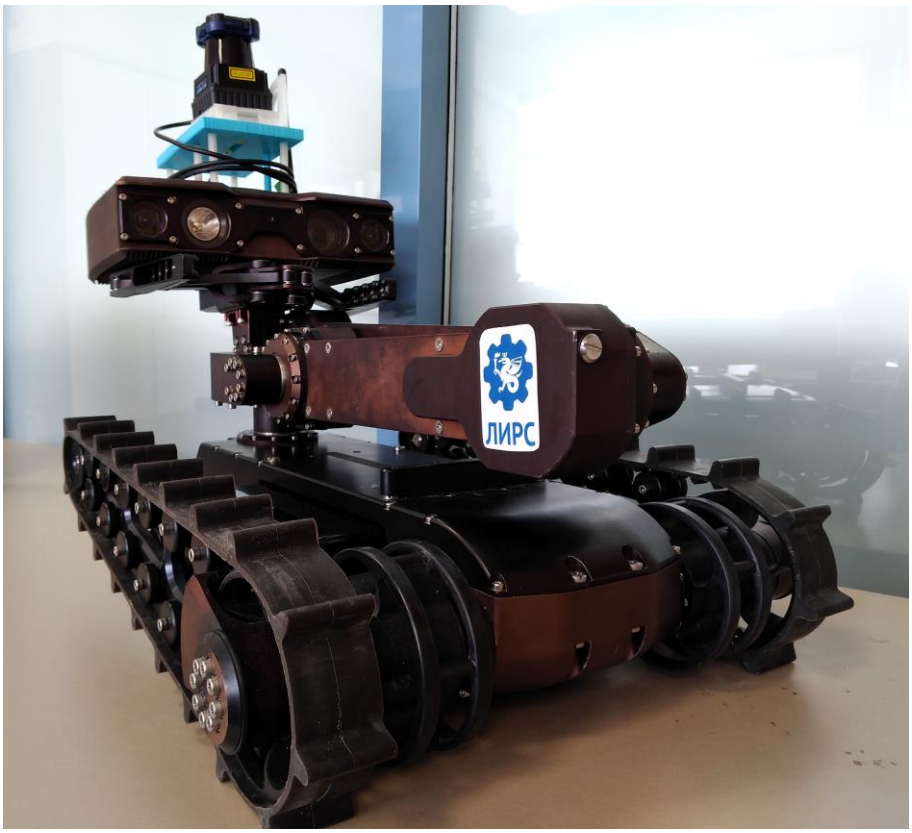

Рисунок 2.1. Робот “Сервосила Инженер” (со старой версией подставки).

“Сервосила Инженер” — это российский гусеничный робот (рис. 2.1) компании “Сервосила” [33]. Робот был сконструирован для работы в поисково-спасательных операциях, и имеет водонепроницаемый металлический корпус. Благодаря тому, что все вычислительные мощности и сенсоры находятся в голове робота, он может преодолевать небольшие водоемы и каналы, заполненные водой. Для этого достаточно поднять голову с помощью манипулятора. В основании робота находится его аккумулятор, благодаря весу которого, несмотря на массивную голову, робот сохраняет устойчивость. Общая масса робота составляет около 17 килограммов. Самой массивной является база робота, содержащая в себе аккумуляторную батарею. Однако при всем этом вес робота достаточно небольшой для того, чтобы его можно было поднять в

одиночку. В таблице 2.1 более детально представлены массовые характеристики робота.

Элемент	Масса
База робота с аккумулятором и поворотными гусеницами	8.80 кг
Манипулятор	4.35 кг
Захват	1.40 кг
Голова робота с бортовым компьютером	2.40 кг
Всего	16,95 кг

Таблица 2.1. Массовые характеристики робота “Сервосила Инженер”.

Робот “Сервосила Инженер” оснащен четырьмя камерами: стерео парой, камерой с оптическим зумом, а также камерой заднего вида. Наличие камеры заднего вида позволяет постоянно оценивать обстановку вокруг робота без необходимости вращения головой. Также робот оснащен фонарем, который может служить источником света в плохо освещенных пространствах или ночью. Это значительно упрощает работу с данным роботом в режиме телеоперации.

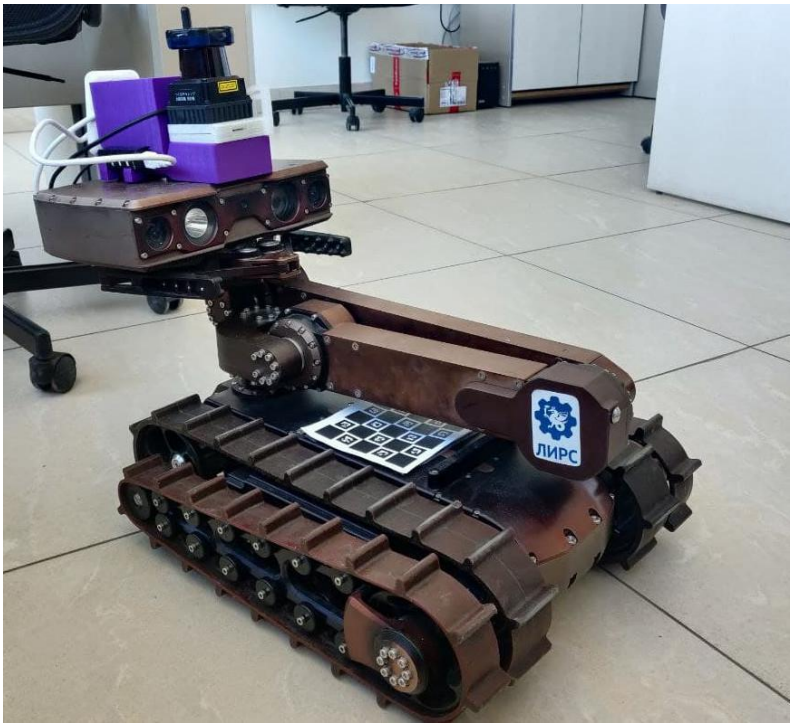

Рисунок 2.2. Робот “Сервосила Инженер” (с новой версией подставки).

Робот также оснащен несколькими датчиками, среди которых есть IMU (инерциальный датчик) и лидар, который расположен на голове робота. На рисунках 2.1-2.2 представлен робот “Сервосила Инженер” с лидаром, установленным на двух различных версиях подставок. Датчик IMU дает информацию о скорости и ускорении робота, а также о его ориентации в пространстве. Лидар измеряет расстояние до объектов, окружающих робота. Для этого используются направленные лучи света, которые отражаясь возвращаются к устройству, благодаря чему измеряется расстояние до препятствий. Набор описанных датчиков является достаточным минимумом для работы робота в автономном режиме [24].

Робот "Сервосила Инженер" имеет пару активных флипперов (дополнительных гусениц), что значительно увеличивает проходимость робота и позволяет ему без труда преодолевать лестницы, а также подниматься и спускаться по отвесным склонам. У робота имеется манипулятор с четырьмя степенями свободы и захватом на конце, что позволяет ему взаимодействовать с

окружающей средой. Голова робота находится на подвижном конце манипулятора, что позволяет роботу заглядывать за препятствия или в окна.

Все вышеобозначенное делает робота "Сервосила Инженер" прекрасной платформой для изучения и реализации как алгоритмов компьютерного зрения, так и различные алгоритмы одновременного картографирования и локализации (SLAM) [19]. А создание симуляционной модели данного робота, сделает процесс реализации и интеграции этих алгоритмов проще и безопасней.

ЧАСТЬ 3. РАЗРАБОТКА СИМУЛЯЦИОННОЙ МОДЕЛИ РОБОТА "СЕРВОСИЛА ИНЖЕНЕР"

3.1. Исходная симуляционная модель робота “Сервосила Инженер”

В данном разделе представлены результаты предыдущих работ по созданию симуляционной модели робота "Сервосила Инженер", некоторые из которых даже стали заделом для будущего исследования.

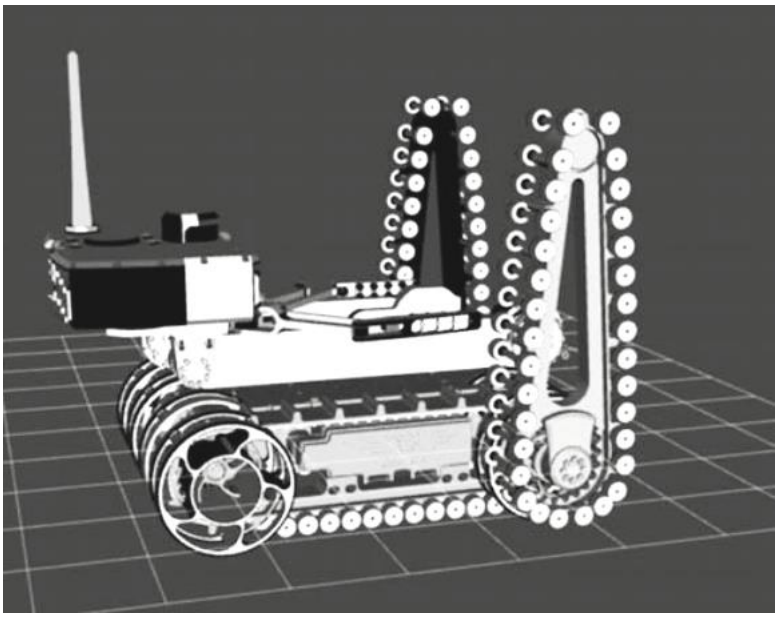

Рисунок 3.1.1. Результат работы по симуляции гусениц.

На рисунке 3.1.1 изображена симуляционная модель робота “Сервосила Инженер”, которая является результатом работы [3] по симуляции гусениц. Стоит отметить, что в результате в симуляции гусениц колесами покрывается не вся рабочая поверхность гусениц, а лишь их часть. В этом случае при перевороте или падении набок часть гусениц будет неактивна, что делает симуляцию недостаточно достоверной.

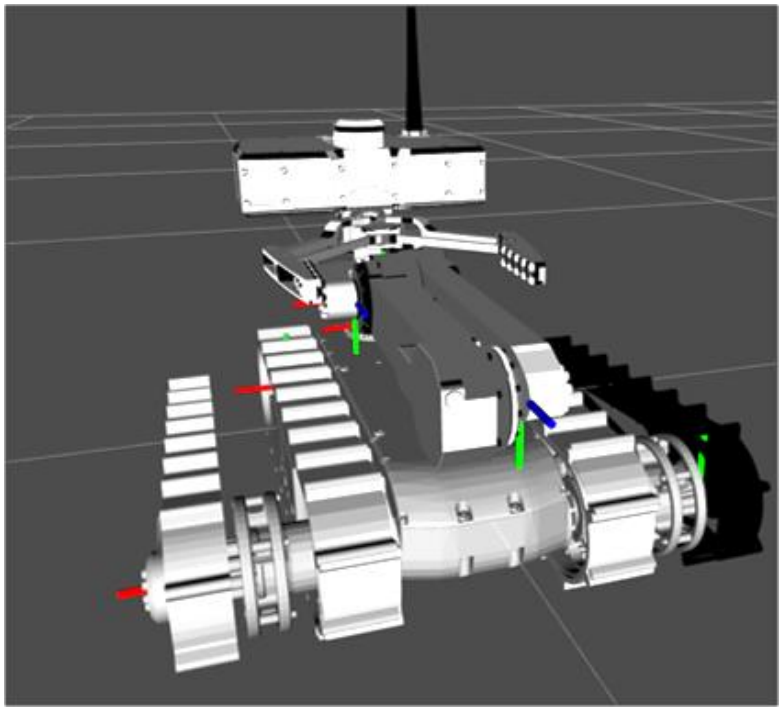

Рисунок 3.1.2. Симуляционная модель робота на начальном этапе работы.

Работа [1] является основополагающей в теме симуляции самого робота. На рисунке 3.1.2 представлена финальная версия модели робота, которая была получена в результате этого исследования. Среди ее недостатков можно отметить несоответствующее действительности положение флипперов, вывернутые нормали левого флиппера (на рисунке он изображен черным цветом). Также стоит отметить, что у данной модели отсутствует подвижный захват — он соединен с головой робота в одну цельную модель. Модель также не имеет лидара, установленного на голове настоящего робота. У модели также полностью отсутствуют сенсоры и камеры, присутствующие на реальном роботе.

Модель, представленная на рисунке 3.1.2, была использована в качестве базы при добавлении дальнейших изменений и улучшений.

3.2. Изменения в симуляционной модели робота "Сервосила Инженер" в симуляторе Gazebo

Для того, чтобы изменить или дополнить модель в симуляторе Gazebo или создать ее с нуля, ROS использует особый формат файлов URDF. Данные файлы представляют собой детальное описание робота, а именно всех его элементов, а также связей между ними. Для обозначения каждого из элементов есть свои теги, так например:

- <link> — тег для описания элементов робота, обязательно содержание информации о визуальном представлении, о модели для просчета коллизий, а также инерциальном блоке, соответствующем описанному элементу.
- <joint> — тег для описания сочленений робота, обязательно должен содержать информацию об элементах, которые соединяются (родительский и дочерний), тип сочленения (фиксированный, вращающийся или призматический), в зависимости от которого также может требоваться информация об ограничениях в движении, силе трения или максимальной скорости движения.
- <gazebo> — специализированный тег для обозначения дополнительной информации, необходимой для симуляции; с помощью данного тега могут вызываться плагины сенсоров, уточняться параметры трения и качества взаимодействия элементов робота между собой.

Примеры фрагментов URDF-файлов можно найти в приложении.

3.2.1. Модели коллизий

В первую очередь при работе с моделью робота “Сервосила Инженер” были изучены модели коллизий. Модели коллизий представляют собой упрощенные модели объекта с сохранением максимально необходимой геометрической формы. Эти модели используются для расчета физики взаимодействия объектов

между собой. Такой подход практикуется при условии использования сложных моделей в симуляции, как например в случае с роботом “Сервосила Инженер”, где модель одной головы имеет почти сто тысяч полигонов. В случае таких сложных моделей расчет физики взаимодействия с окружающей средой может занять большое количество времени. Тогда как модель коллизий зачастую сохраняет лишь общий силуэт объекта и значительно упрощает вычисления.

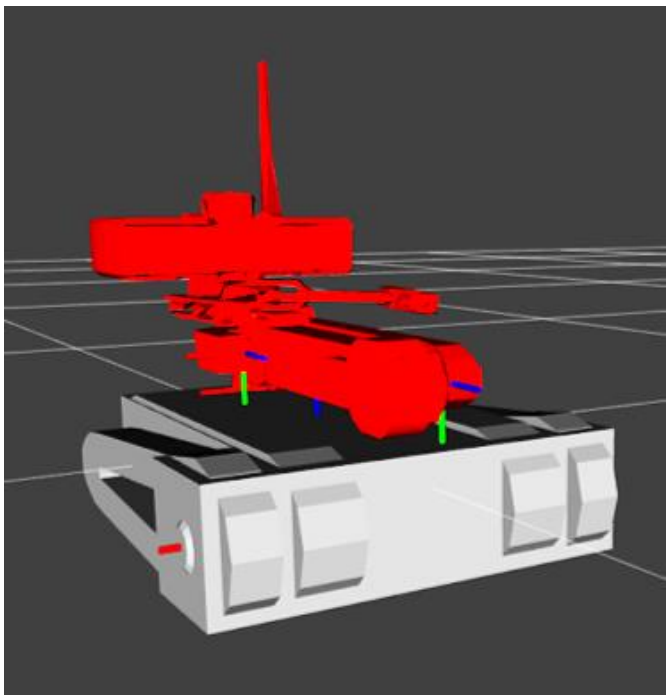

Рисунок 3.2.1.1. Исходные меши коллизий робота (отмечены красным).

В случае существующей модели для создания моделей коллизий был использован автоматический генератор, который алгоритмически снижает количество полигонов, стараясь сохранять общую геометрию модели. К сожалению, при подобном подходе модели коллизий получаются ломаными. Генератор зачастую не способен понять, насколько важны те или иные элементы упрощаемого объекта, насколько сильно и в каких местах их можно упростить. На рисунке 3.2.1.1 в красном цвете представлены исходные модели коллизий манипулятора робота.

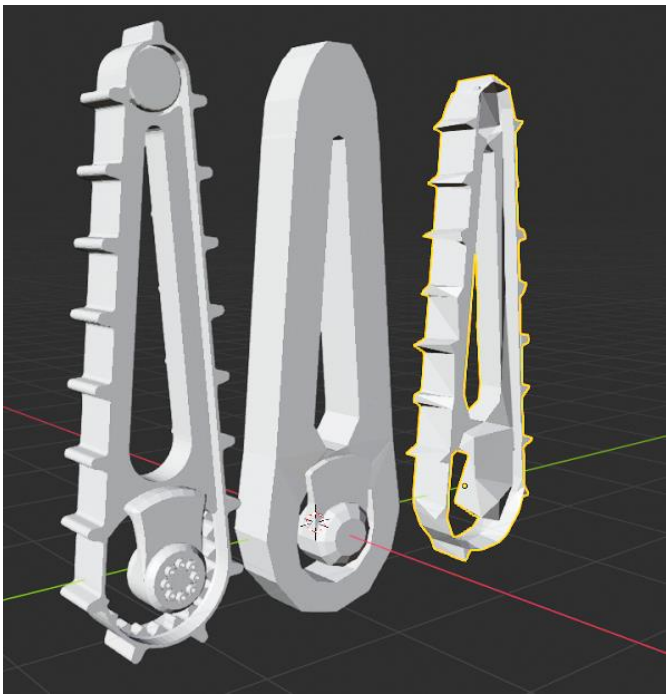

Рисунок 3.2.1.2. Сравнение моделей флиппера (слева — визуальная модель, справа — автоматически сгенерированная модель коллизий, по центру — исправленная модель коллизий)

Исправление автоматически сгенерированных моделей коллизий стало первым этапом в доработке симуляционной модели робота “Сервосила Инженер”. На рисунке 3.2.1.2 представлено сравнение разных моделей флиппера. Слева на рисунке представлена исходная визуальная модель флиппера, справа изображена автоматически сгенерированная модель флиппера, для использования в качестве модели для просчета коллизий. В центре представлена созданная с нуля коллизионная модель для флиппера. Можно заметить разительные отличия между автоматически сгенерированной моделью коллизий и моделью, созданной с нуля. Так автоматически сгенерированная модель имеет неравномерную структуру гусениц, которые в большинстве своем превратились в зубцы разной формы. Основное крепление флиппера также потеряло свою форму, в итоговом виде представляя собой выпуклость пирамидальной формы. Средняя модель представляет собой аккуратно

упрощенную форму флиппера. Форма внешней стороны огибает гусеничный ряд, не воссоздавая пустое пространство между зубцами, лишняя детализация в виде болтов, гаек и фасок также была устранена.

Схожим образом были созданы модели всех элементов робота. Большие изменения коснулись только модели головы. Если раньше она представляла собой цельную конструкцию из головы и захвата, то после внесенных изменений голова и захват стали отдельными моделями. Также была добавлена модель лидара и подставки. Для конкретизации и замены моделей коллизии в URDF с описанием робота менялось наполнение тегов <collision> в тегах <link>.

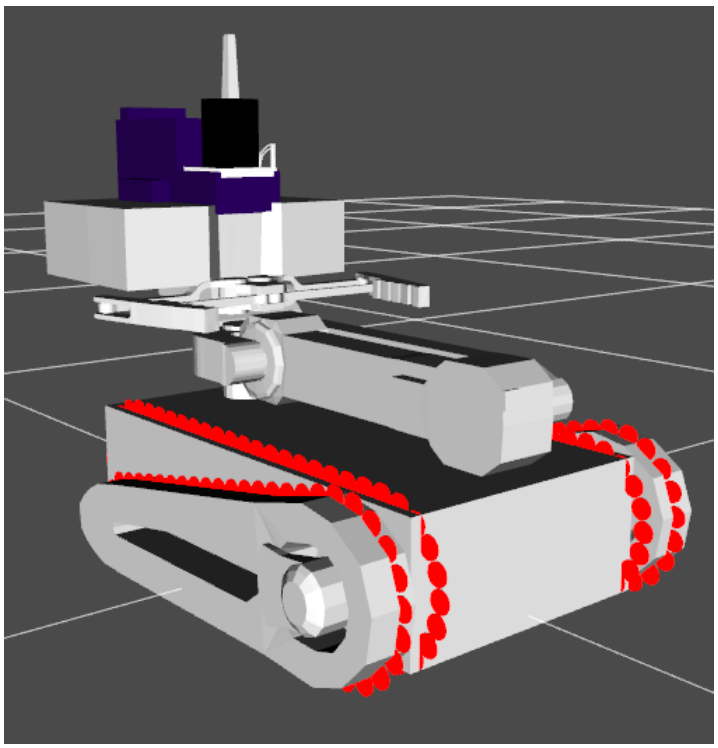

Рисунок 3.2.1.3. Исправленные модели коллизий робота “Сервосила Инженер”.

Финальное отображение моделей коллизий всего робота представлено на рисунке 3.2.1.3. Можно заметить описанные изменения в модели: разделение модели головы и захвата, добавление лидара и подставки под него.

Для сравнения объемов получившихся моделей ниже представлена таблица 3.2.1.1 со сравнением количества полигонов в различных версиях модели робота. Стоит отметить, что значение RTF снято в положении покоя модели, то есть взаимодействие модели с окружающим миром ограничивается лишь столкновением основания базы робота и пола в симуляции. Также можно отметить небольшое снижения количества полигонов, даже несмотря на увеличение числа деталей, использованных в модели робота.

Модели	Визуальные модели	Автоматически сгенерированные модели коллизий	**Новые модели коллизий**
Количество отдельных фрагментов в модели	15	9	**15**
Количество полигонов	967 219	5 712	**5 644**
RTF (Real Time Factor)	~ 0.2	~ 1	**~ 1**

Таблица 3.2.1.1. Сравнение моделей робота.

3.2.2. Визуальные модели

Следующим изменения коснулись визуальной составляющей робота. Флипперы, ранее смещенные относительно точки крепления к базе робота, были установлены на места крепления аналогично реальному роботу, а нормали одного из флипперов были–исправлены. Модель была затекстурирована в соответствие с реальным роботом для большей схожести симуляционной модели с реальностью. Итоговый внешний вид модели представлен на рисунке 3.2.2.1.

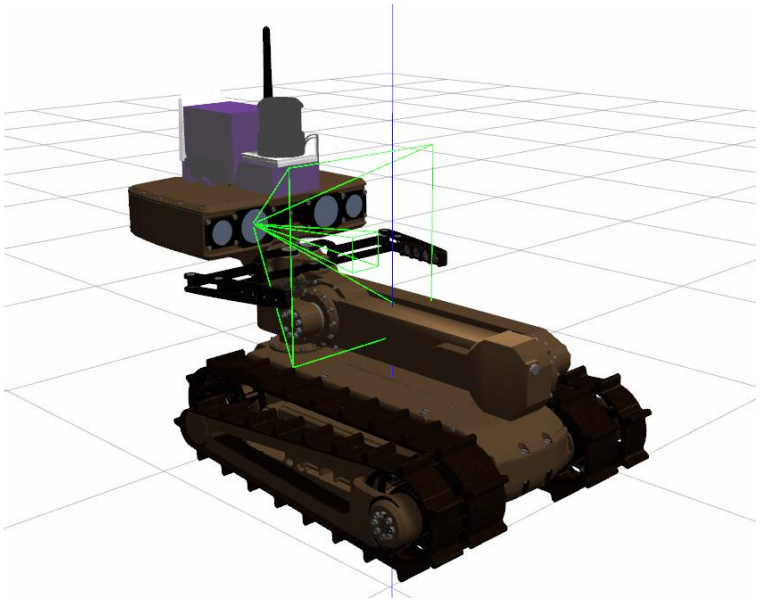

Рисунок 3.2.2.1. Изображение модели робота “Сервосила Инженер” после внесения визуальных правок.

3.2.3. Конструкция модели робота

Как упоминалось ранее, конструкция робота также подверглась некоторым изменениям. У симуляционной модели появился активный захват. Для его создания в первую очередь была разделена на составляющие модель головы робота. Сам захват состоит из шести частей – по три с каждой из сторон. Поэтому для корректного движения захвата в симуляции все шесть частей должны двигаться синхронно. На рисунке 3.2.3.1 представлена схема захвата, содержащая отметки о соотношении углов поворота каждого из его элементов при движении. Так, для открытия захвата на угол β необходимо повернуть две части одной половины захвата идущие от головы на угол β, а третью часть захвата, соединяющие первые две повернуть на угол -β. Вторая половина захвата должна двигаться по идентичной схеме, поворачивая в противоположную сторону, то есть первые две части поворачивают на угол -β, а третья – на угол β.

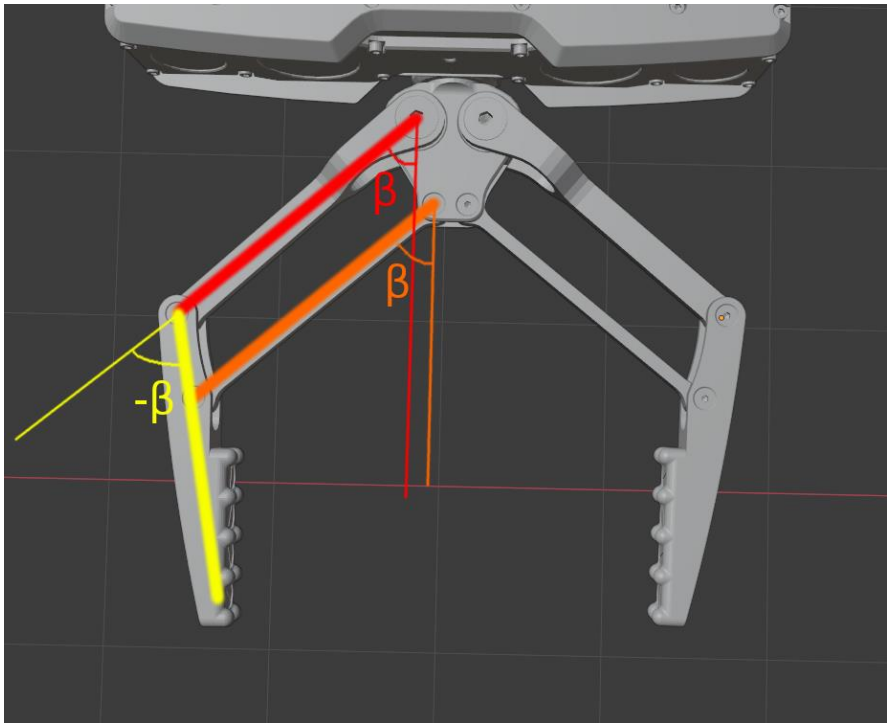

Рисунок 3.2.3.1. Схема захвата с углами поворота его частей.

Для описания захвата и его конфигурации был создан новый URDF файл, фрагменты которого представлены в приложении. Ссылка на добавленный файл описания также содержится в файле с описанием манипулятора. Реализация синхронного движения была обеспечена тегом <mimic> в описании сочленений. Данный тег позволяет подвижным соединениям копировать поведение друг друга. В описанном выше случае, только одно соединение было описано как самостоятельное, а именно сочленение в основании первой части правой половины захвата. Остальные пять соединений были помечены тегом <mimic> и в разном соответствии копируют поведение первого. Подобная схема привела к тому, что управление всем захватом происходит с помощью контроля одного единственного соединения из шести имеющихся в захвате.

Рисунок 3.2.3.2. Подставка под лазерный дальномер на реальном роботе.

Также к симуляционной модели был добавлен лидар и его подставка. На рисунке 3.2.3.2 представлен внешний вид подставки под лидар, расположенной на голове реального робота. Подставка была распечатана на 3D принтере, что подразумевает наличие моделей, на базе которых производилась печать.

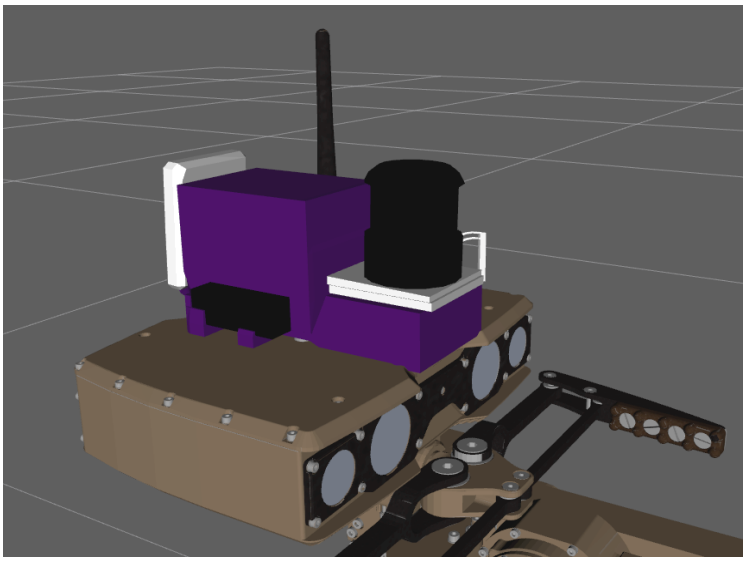

Рисунок 3.2.3.3. Подставка под лазерный дальномер на модели робота “Сервосила Инженер”

Используя модели, которые ранее были использованы для печати новой подставки, была создана упрощенная модель данной подставки в симуляции. Симуляционная модель подставки содержит визуальное наполнение идентичное настоящему: черный параллелепипед, который моделирует USB-Hub, и белый параллелепипед, моделирующий Wi-Fi адаптер. Рисунок 3.2.3.3 иллюстрирует итоговый внешний вид поставки.

Описанные элементы были добавлены в URDF файл, содержащий описание манипулятора, с помощью добавления в описание элементов модели. Модель добавила роботу три новых элемента:

- Основная часть подставки, которая устанавливается на голову робота, с визуальным наполнением.
- Дополнительная часть подставки, к которой на реальном роботе крепится сам лидар. Она представлена отдельным элементом для сохранения возможности изменения положения лидара относительно головы робота, которая имеется у реальной подставки.
- Модель лидара. Данный элемент был представлен в качестве отдельного, так как именно к нему далее будет прикрепляться плагин лазерного дальномера, используемый для симуляции поведения лидара в симуляторе Gazebo.

Таким образом, в конструкцию симуляционной модели робота “Сервосила Инженер” были добавлены активный захват, а также модель подставки под лидар и модель самого лидара.

3.2.4. Симуляция гусениц

При симуляции гусениц было решено опираться на работу [3], которая содержит в себе основную информацию касательно этого вопроса. Потому было решено использовать по 56 колес для симуляции всей поверхности каждой из двух основных гусениц.

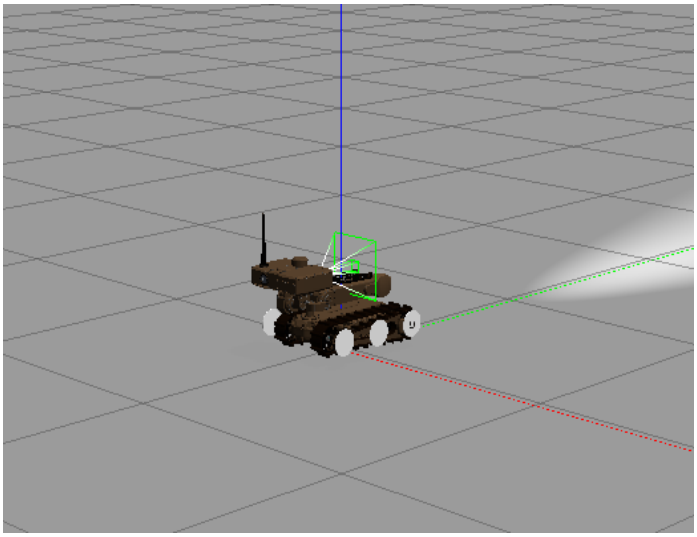

Рисунок 3.2.4.1. Изначальный внешний вид колес для симуляции основных гусениц.

На момент начала работы в симуляции гусениц модели робота «Сервосила Инженера» имелось шесть крупных колес (по три на каждую гусеницу) под управлением differential_drive контроллера, который позволяет управлять всеми колесами сразу и придавать им определенную скорость, а также угол поворота. Однако настройки контроллера были неправильными из-за чего модель вела себя некорректно; например, скорость движения робота была очень низкой и не соответствовала скорости вращения колес. Изначальный вид колес для симуляции представлен на рисунке 3.2.4.1.

Таким образом, в рамках обозначенной задачи касательно основных гусениц речь шла об увеличении количества колес и соответственно уменьшении их размера, а также подстройке контроллера. Симуляция гусениц на флипперах была осуществлена по тому же принципу – тот же размер колес и управление с помощью differential_drive контроллера. Соответствуя получившемуся размеру колес на основных гусеницах, для покрытия всей рабочей поверхности гусениц флипперов понадобилось по 50 колес на каждый из флипперов. После настройки контроллеров управление гусеницами стало

осуществляться с помощью ROS топика /cmd_vel с сообщениями типа Twist, в которых может указываться линейная и угловая скорости робота.

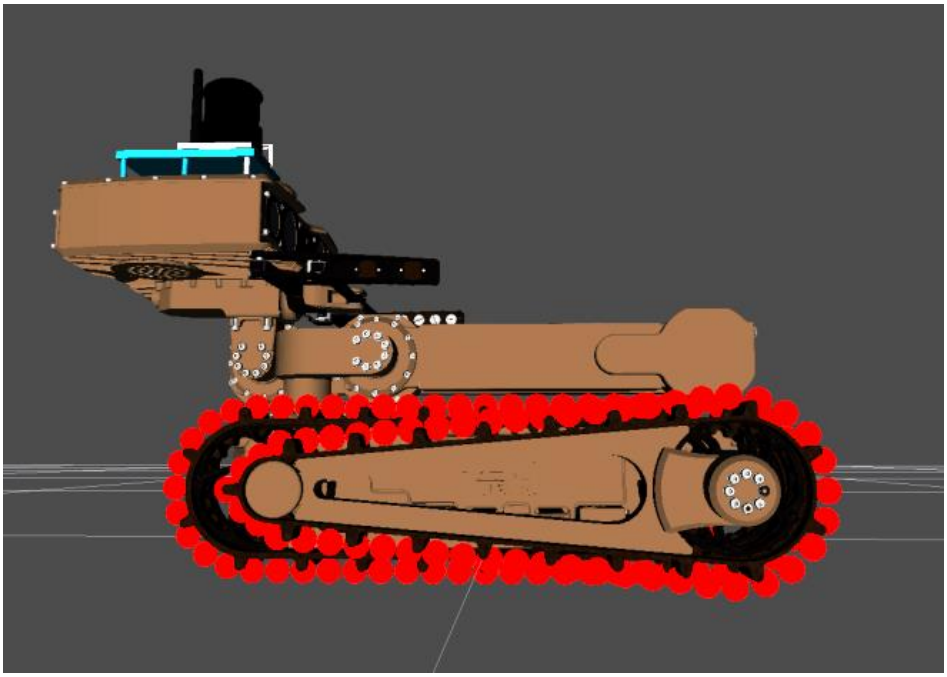

Рисунок 3.2.4.2. Финальное изображение колес для симуляции гусениц.

На рисунке 3.2.4.2 изображено финальное расположение симуляционных колес на модели робота “Сервосила Инженер”. Симуляционными колесами покрыты все поверхности гусениц, что обеспечивает максимальную идентичность поведению гусениц на реальном роботе. Суммарно для симуляции всех гусениц робота было использовано 212 колес.

3.2.5. Сенсоры и камеры

Следующим этапом работы над симуляционной моделью робота “Сервосила Инженер” стало добавление сенсоров и контроллеров. Робот “Сервосила Инженер” оснащен следующими сенсорами и инструментами:

- Камеры:
    - Стерео пара
    - Камера с оптическим зумом
    - Задняя камера

- Инерциальный датчик (IMU)
- Лазерный дальномер (лидар)
- Фонарик

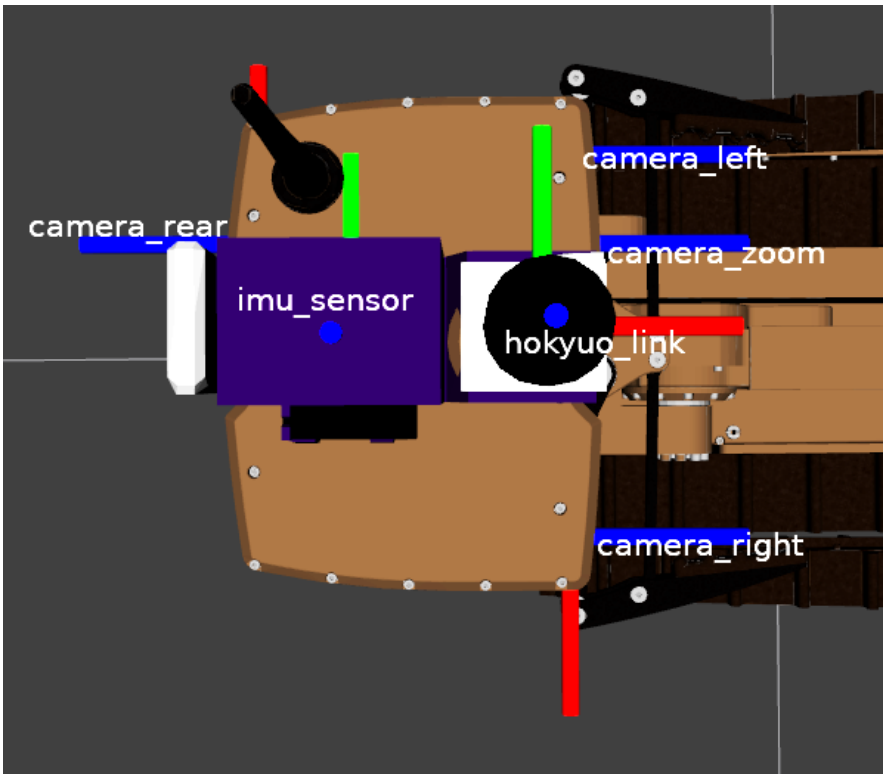

Рисунок 3.2.5.1. Расположение сенсоров и камер на голове симуляционной модели робота.

Добавление сенсоров к модели производится с помощью подключения соответствующих плагинов Gazebo к элементам модели с помощью тега <gazebo> в URDF файле. На рисунке 3.2.5.1 представлены расположения элементов робота, к которым прикреплены соответствующие их названиям сенсоры и камеры. Далее будут более конкретно рассмотрены данные, с каждого из сенсоров.

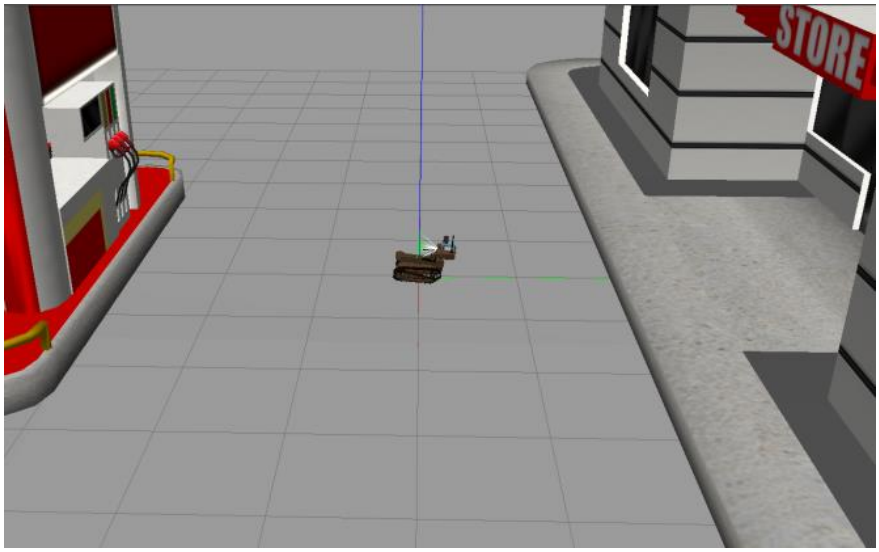

Рисунок 3.2.5.2. Положение робота в симуляционной среде.

Первыми к симуляционной модели были добавлены камеры – три спереди и одна сзади. Для проверки работы добавленных в описание робота камер, модель робота была запущена в симуляционную среду с установленной моделью заправки, которая является шаблонным элементом, по умолчанию доступным в симуляторе Gazebo (рисунок 3.2.5.2).

Рисунок 3.2.5.3 иллюстрирует данные с камер, полученные с робота в данном положении. Два верхних изображения приходят с камер, представляющих стерео пару реального робота. Левое нижнее изображение предоставляет данные с камеры, соответствующей камере с оптическим зумом на реальном роботе, а правое нижнее изображение предоставляет данные с камеры заднего вида. Данные с камер подтверждают, что они были удачно интегрированы в симуляционную модель робота “Сервосила Инженер”.

Рисунок 3.2.5.3. Изображения, поступающие с камер, установленных на роботе.

Следующим этапом добавления сенсоров в симуляционную модель стало добавление IMU сенсора и лидара. Процесс их подключения к модели пришел к добавлению плагинов соответствующих сенсоров с тегом <gazebo> в URDF файл с описанием манипулятора. Для того, чтобы проверить их работу, модель робота была помещена в симуляцию, наполненную примитивными фигурами (кубы, цилиндры, шары; рисунок 3.2.5.4).

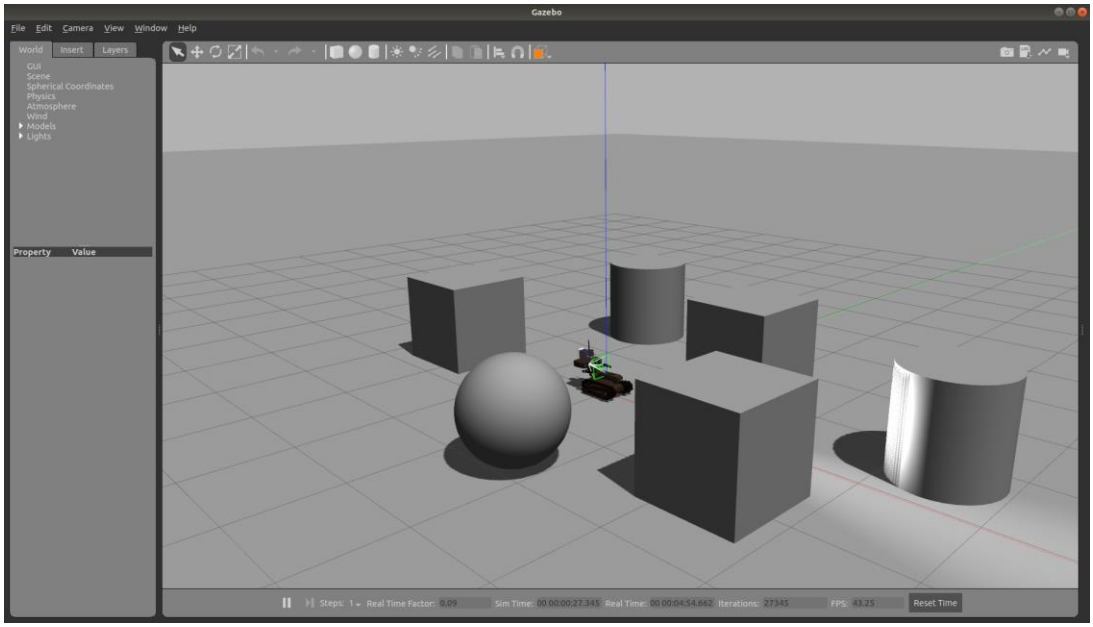

Рисунок 3.2.5.4. Положение робота в симуляционной среде.

Для визуализации данных, полученных с проверяемых датчиков (IMU и лидара), параллельно с вышеописанной симуляцией был запущен RViz. С помощью него удалось четко увидеть полученную с датчиков информацию. На рисунке 3.2.5.5 представлена визуализация данных с лидара и IMU датчика в визуализаторе RViz. Данные с IMU отображаются в виде красной стрелки, исходящей из положения сенсора и показывающей ориентацию робота относительно подстилающей поверхности симуляции. Массив красных точек представляет данные, получаемые с лидара. Линии, получаемые благодаря представленному массиву точек, в точности повторяют границы предметов, перед которыми находится модель робота в симуляции.

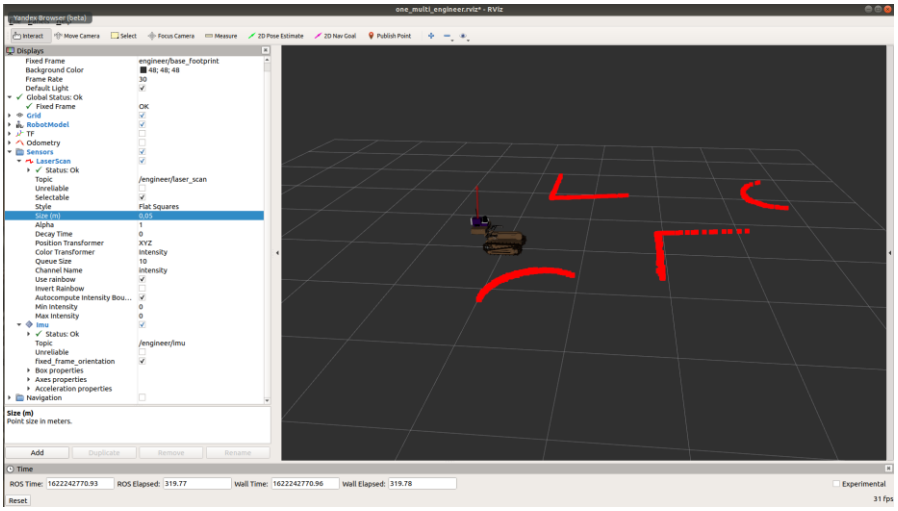

Рисунок 3.2.5.5. Данные полученные с лидара и IMU датчика, отображенные в RViz.

Добавление фонаря стало заключительным этапом работы с симуляцией оборудования робота. Одним из решений задачи его симуляции стал плагин Flash Light для Gazebo, который использовался для прикрепления источника света к объектам. С его помощью можно создать источник света отдельно от робота, описав его в отдельном файле с описанием симуляционной среды Gazebo (world файл). В конечном итоге этот способ представлялся не очень удобным, так как если появится необходимость переместить робота в новый симуляционный мир, то перенос будет затрагивать не просто копирование ссылки на URDF модель робота, но и вмешательство в файл конфигурации симуляционного мира. Решением стало добавление источника света (типа “точечный свет”, который по внешнему виду соответствует реальным фонарям) в описание робота. А реализация его выключения и включения осуществляется путем вызова сервиса /set_light_properties, для изменения состояния источника света в симуляторе Gazebo. Результат работы фонарика можно увидеть на рисунке 3.2.5.4.

3.2.6. Инерции

Исправление инерций модели робота стала одной из ключевых задач при работе с симуляционной моделью робота “Сервосила Инженер”. Описание

инерций в условиях симуляции помогает максимально точно описать поведение всей модели и отдельных ее частей в условиях взаимодействия друг с другом и окружающей средой.

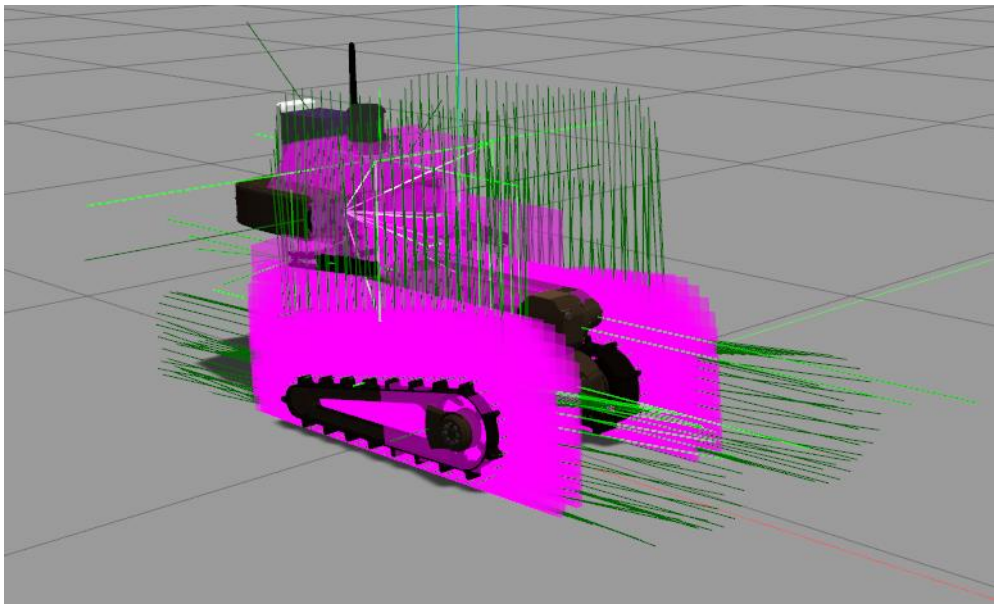

Рисунок 3.2.6.1. Исходные инерции модели робота.

На рисунке 3.2.6.1 можно увидеть исходные инерции модели робота “Сервосила Инженер”. Самыми заметными отличительными особенностями этой модели являются большие значения инерций колес, используемых для симуляции гусениц робота, а также вертикально повернутая инерция базы робота. И если в стационарном положении робота большие инерции симуляционных колес практически не влияют на поведение модели, то неверно выставленная инерция базы робота стала причиной определенных сложностей при движении робота.

Будучи повернутой в вертикальное положение, данный инерциальный блок значительно взаимодействует с поверхностью, по которой едет робот, заставляя его “пробуксовывать”. Данная проблема была замечена при работе над стеком навигации робота. “Пробуксовка” робота при движении в условиях одновременной локализации и картографирования пагубно влияла на поведение

алгоритмов навигации, так как создавала критическую ошибку в потоке данных одометрии, приходящих с модели робота. Так, при подаче команды движения модели робота с определенной скоростью, одометрия рассчитывалась из скорости движения колес и создавала некоторое предположительное изменение положения робота в окружающей среде. Однако из-за “пробуксовки” фактическое положение модели сильно отличалось от предполагаемого; более того, с каждой итерацией ошибка накапливалась, быстро достигая критической отметки, что приводило к погрешностям при построении карты и соответственно к ошибкам при планировании пути.

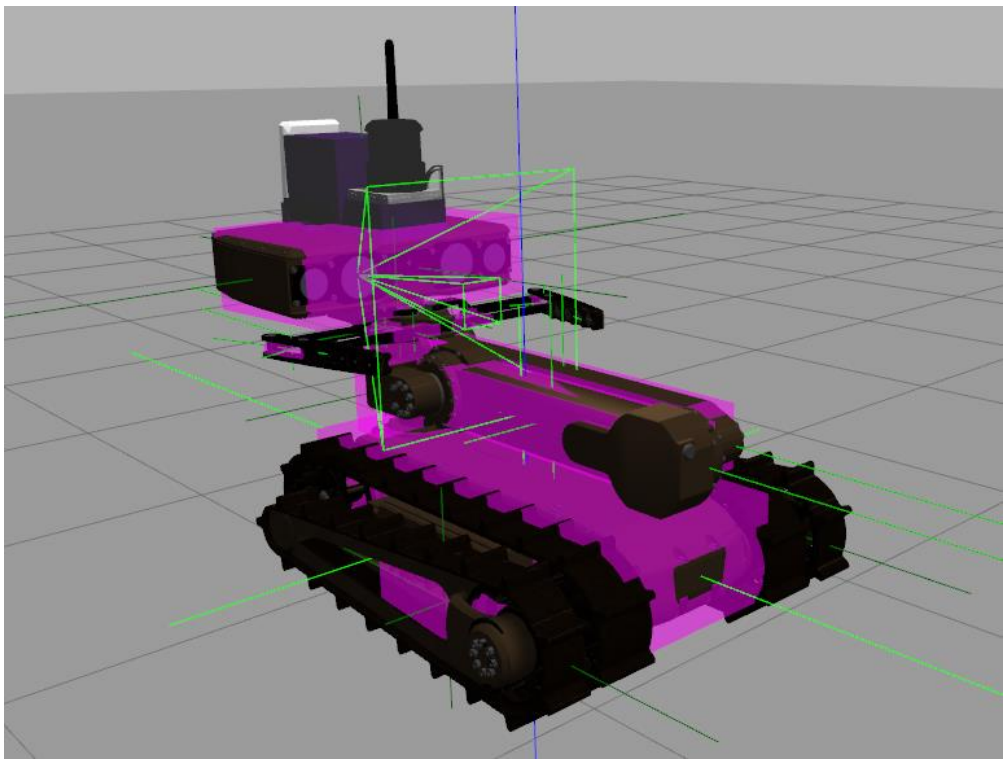

Рисунок 3.2.6.2. Измененные инерции основных элементов модели робота.

Описанная выше проблема была исправлена поворотом инерциального блока базы робота в горизонтальное положение в соответствии с положением самой модели. На рис. 3.2.6.2 можно увидеть внесенные в основное строение робота (база, флипперы, манипулятор) изменения. На данном этапе

симуляционные колеса были отключены для большего удобства и упрощения визуального доступа к остальным инерциям робота.

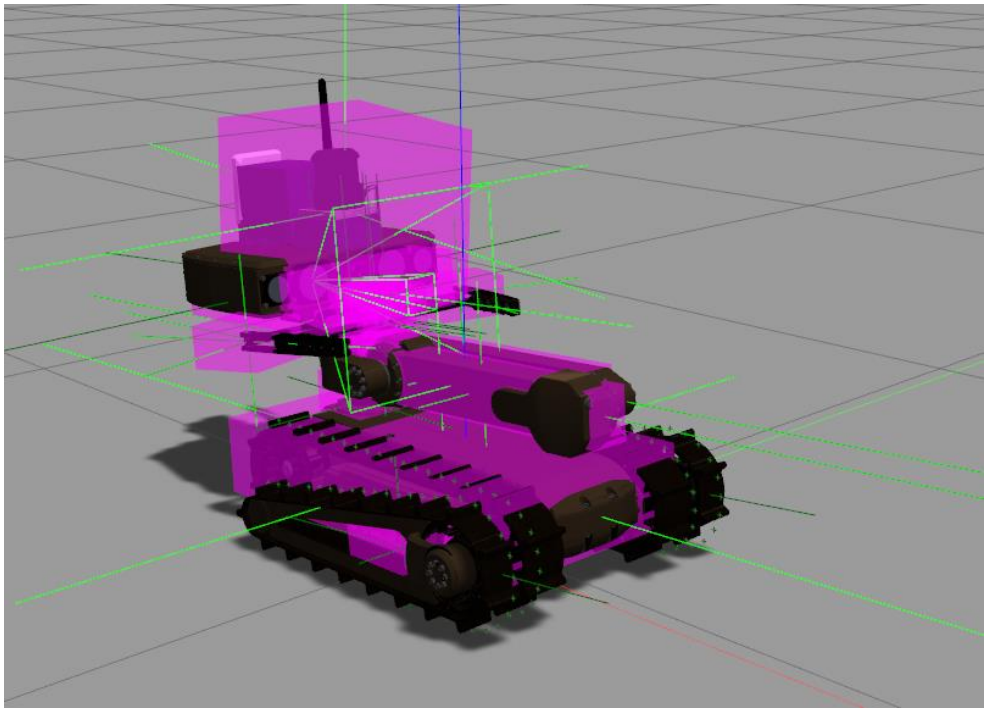

Рисунок 3.2.6.3. Итоговое отображение инерций модели робота.

Следующим этапом исправления инерций робота стало возвращение симуляционных колес и настройка их инерций, а также настройка инерций для подставки для лидара на голове робота. Стоит отметить, что на данном этапе пришлось столкнуться с проблемами в работе симулятора Gazebo, так при установлении подходящих значений инерций модели, при запуске симуляции робот стал разлетаться на части. Достоверно известно, что причиной данной проблемы в симуляторе Gazebo являются неправильные или невоспроизводимые для симулятора значения инерций. Для исправления возникшей проблемы пришлось провести подбор некоторых параметров инерции, а итоговый внешний вид инерций робота изображен на рисунке 3.2.6.3.

3.3. Создание симуляционной модели робота "Сервосила Инженер" в симуляторе Webots

Одной из проблем, с которой пришлось столкнуться при создании симуляционной модели робота “Сервосила Инженер” в симуляторе Gazebo, стало отсутствие в нем необходимого инструментария для симуляции гусениц, двумя парами которых оснащен робот. Для обхода данной проблемы было предложено решение симулировать поведение гусениц с помощью массива виртуальных колес. Для большей достоверности симуляции колеса имели небольшой размер. Таким образом для симуляции всех гусениц робота пришлось использовать более двух сотен колес. Добавление такого количества элементов к симуляционной модели привело к тому, что RTF симуляции снизился более чем в 5 раз – с 0.9-1 до 0,1-0,2. Это говорит о резком снижении скорости симуляции. Также большое количество элементов в модели периодически вызывает разного уровня критичности сбои в работе симулятора.

В качестве альтернативы на следующем этапе работы было решено рассмотреть еще один симулятор – Webots. Данный симулятор показывает неплохие результаты по качеству симуляции физики и, в отличии от Gazebo, имеет полноценный инструментарий для симуляции гусениц [26]. Было решено создать симуляционную модель робота “Сервосила Инженер” в симуляторе Webots.

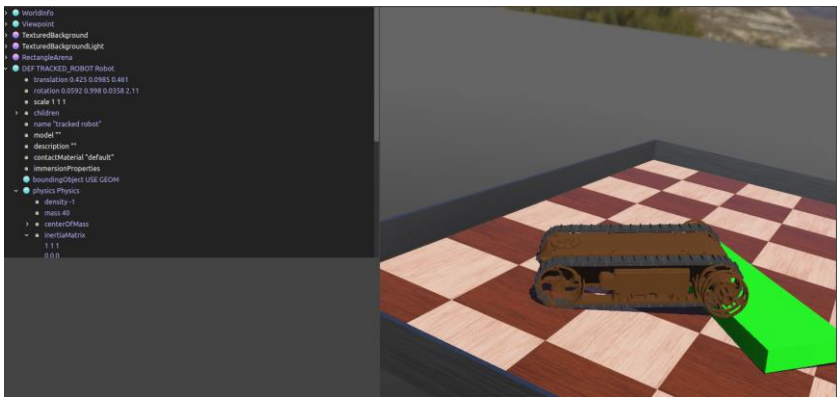

Рисунок 3.3.1. База робота в симуляторе Webots.

Была проведена проверка работы симулятора и симуляции гусениц. На этом этапе была использована лишь модель базы робота и пара гусениц, установленных на ней. Иллюстрация проверки работы симуляции гусениц представлена на рисунке 3.3.1.

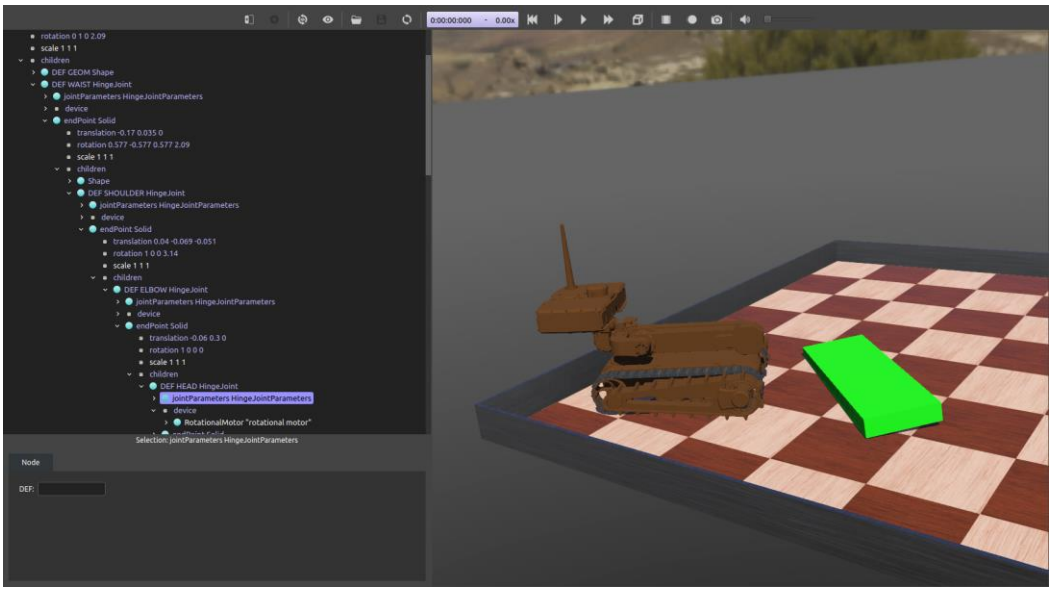

Рисунок 3.3.2. Робот с готовым манипулятором в симуляторе Webots.

Следующим этапом стала сборка оставшейся модели робота. Симулятор Webots имеет совершенно отличный от Gazebo графический интерфейс. И если для создания модели робота в Gazebo необходимо создавать URDF файлы с его описанием, то в Webots создание модели робота идет непосредственно в самом симуляторе. На данный момент модель робота “Сервосила Инженер” в симуляторе Webots имеет следующий внешний вид (рис. 3.3.2).

ЧАСТЬ 4. СТЕК НАВИГАЦИИ

4.1. Добавление стека навигации к симуляционной модели робота "Сервосила Инженер" в симуляторе Gazebo

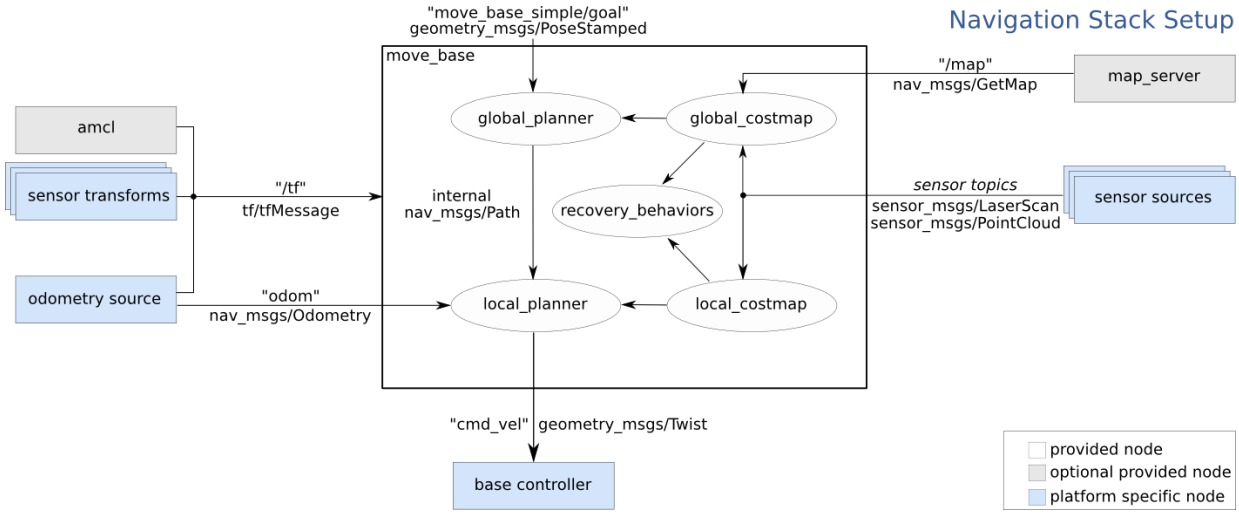

Рисунок 4.1.1. Схема работы стека навигации в ROS.

Затрагивая тему работы и настройки стека навигации в ROS/Gazebo, необходимо описать принцип его работы на роботах, в условиях работы с ROS. Большинство алгоритмов планирования пути и картографирования уже реализованы в робототехнической операционной системе и все они работают примерно по одной и той же схеме, идентичной приведенной на рисунке 4.1.1. Для работы стека навигации ROS использует данные, поступающие с лидара, и одометрию положения робота. Основным в данном случае выступает пакет move_base, он содержит информацию о локальном и глобальном планировщиках и картах стоимости. Отталкиваясь от параметров, заданных в этих модулях, данный пакет строит оптимальный путь, основываясь на имеющейся карте местности.

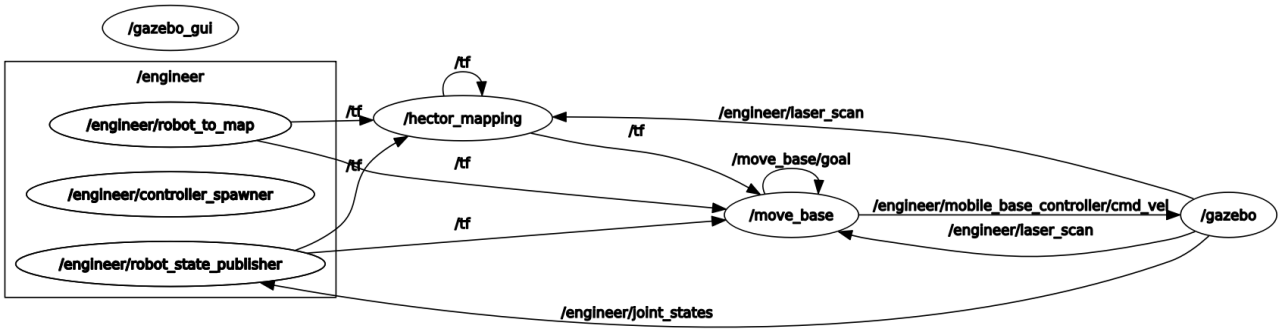

Рисунок 4.1.2. Схема работы стека навигации на симуляционной модели.

Для симуляционной модели робота “Сервосила Инженер” было решено использовать стек навигации идентичный тому, что используется на реальном роботе. Он использует алгоритм hector_mapping для задачи одновременного картографирования и локализации (SLAM). В качестве локального планировщика используется dwa_planner. Для использования его на симуляционной модели были произведены небольшие изменения в наименованиях элементов модели, отвечающих за передачу данных с сенсоров, а также отвечающих за положение робота в симуляции. Итоговая схема работы стека навигации в симуляции представлена на рисунке 4.1.2.

4.2. Эксперименты со стеком навигации

4.2.1. Реальный робот

Для настройки стека навигации было решено провести серию экспериментов на реальном роботе. В настройке навигации помогла работа [30], где четко описаны роли и влияние каждого из его параметров. В приложении В можно найти итоговые параметры стека навигации.

Рисунок 4.2.1.1. Робот в процессе эксперимента со стеком навигации.

В процессе эксперимента со стеком навигации на реальном роботе, робот перемещался по аудитории, используя стек навигации, и строил карту [31]. На рисунке 4.2.1.1 изображен робот в процессе движения по аудитории.

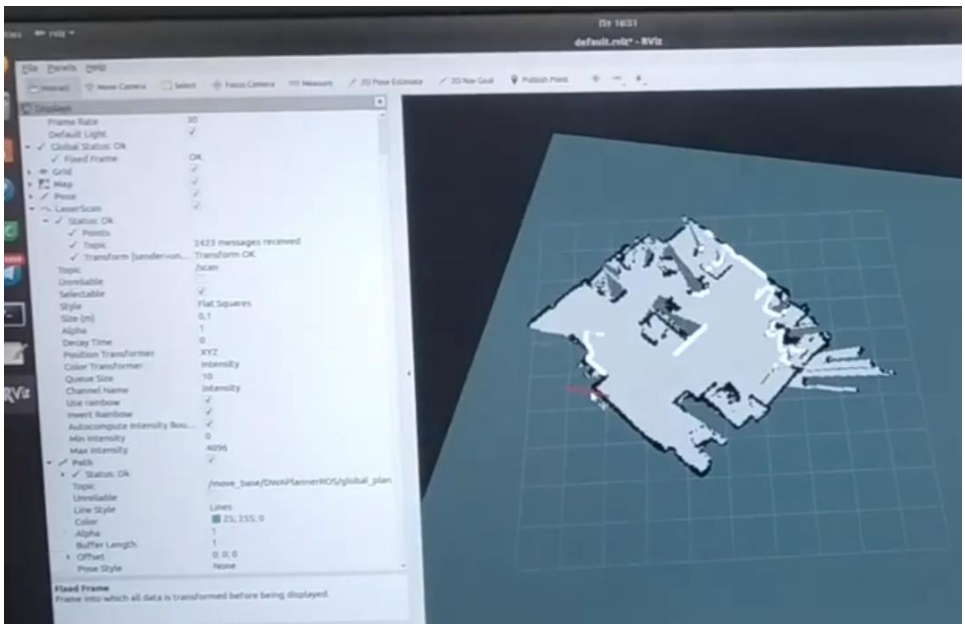

Рисунок 4.2.1.2. Карта, построенная роботом в процессе эксперимента.

Рисунок 4.2.1.2 иллюстрирует карту местности, построенную роботом за время эксперимента. Стек навигации на реальном роботе также реализован с помощью ROS, поэтому для отображения готовой карты и визуализации данных с лазерного дальномера в эксперименте использовался RViz.

4.2.2. Симуляционная модель в Gazebo

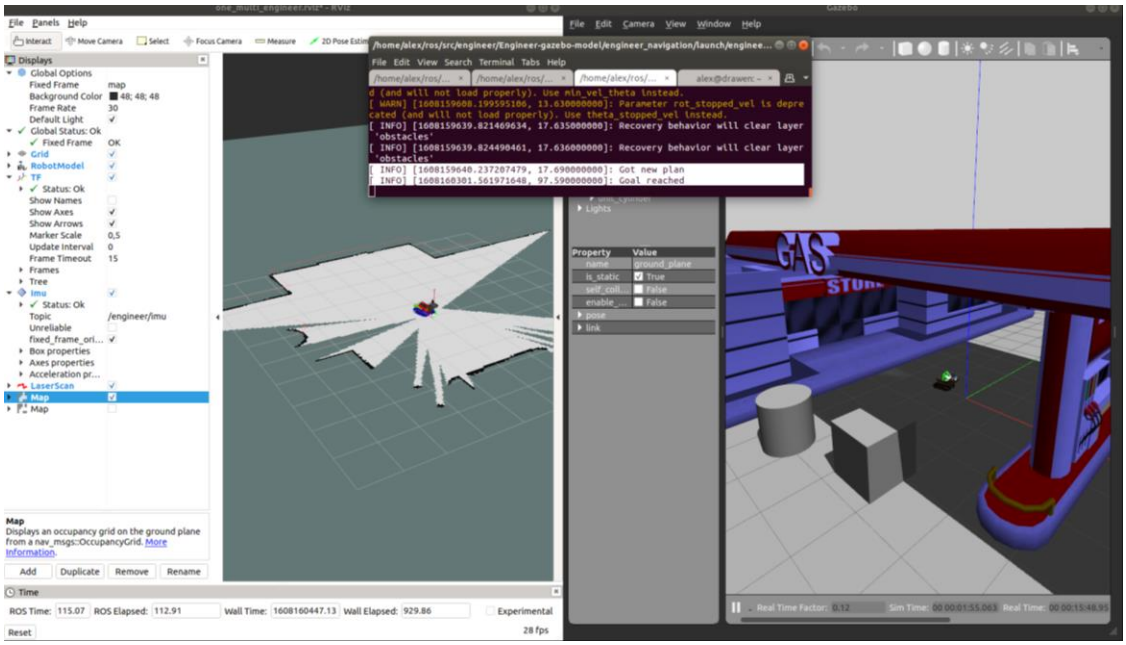

Рисунок 4.2.2.1. Эксперимент по картографированию местности в симуляторе Gazebo (справа – положение робота в симуляторе Gazebo, слева – данные с датчиков робота в визуализаторе RViz).

Когда стек навигации был настроен, все параметры планировщика были перенесены на стек навигации симуляционной модели робота. Для проверки его работы было проведено несколько экспериментов. Первый эксперимент, проиллюстрированный на рисунке 4.2.2.1, представляет результат картографирования местности в симуляторе. На рисунке можно увидеть робота в симуляционной среде Gazebo в мире заправочной станции, а также изображение окна RViz, в котором отображается карта местности, построенная роботом за время эксперимента.

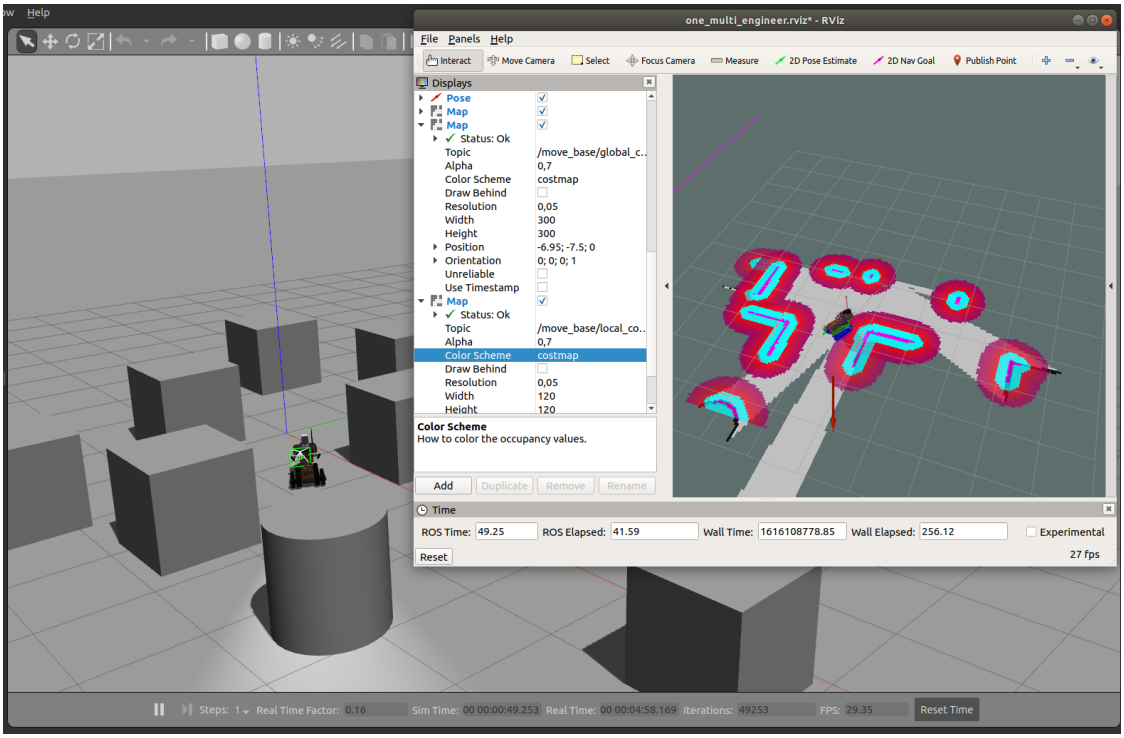

Рисунок 4.2.2.2. Эксперимент, посвященный работе навигации модели робота в симуляторе Gazebo.

Следующий эксперимент был реализован для иллюстрации алгоритма навигации. На рисунке 4.2.2.2 можно увидеть один из этапов работы стека навигации, а именно построение пути и движение по нему на построенной карте местности. Модель робота была добавлена в симуляционную среду, наполненную препятствиями для робота, выполненными из стандартных примитивов (кубов и цилиндров). По мере движения робота происходит построение карты, что также проиллюстрировано в данном эксперименте в окне визуализатора RViz. На рисунке можно увидеть навигационную цель, поставленную оператором (красная стрелка), в направлении которой движется робот. Яркими цветами вокруг препятствий отображается карта стоимости, которая используется для определения границ препятствий для робота, а также дает роботу информацию о том, насколько близко можно подъехать к тому или иному препятствию. Чем ближе цвет к фиолетовому оттенку, тем опасней нахождения даже части робота в нем. В связи с этим алгоритм планировщика

рассчитывает траектории с минимальным вхождением ярких участков карт стоимости. Благодаря картам стоимости робот может понять, что некоторый проход на карте будет слишком узок для габаритов робота. В таком случае планировщик не будет предлагать пути, проложенные через такие проходы.

Проведенные эксперименты показали, что стек навигаций настроен оптимально и успешно работает как в симуляции, так и на реальном роботе.

ЧАСТЬ 5. ДВИЖЕНИЕ ПО НЕРОВНЫМ ПОВЕРХНОСТЯМ

5.1. Создание симуляционной модели RSE в симуляторе Gazebo

Рисунок 5.1.1. Random Step Environment.

Для дальнейшего проведения экспериментов в условиях неровных поверхностей как в симуляции, так и на реальном роботе было решено воссоздать имеющийся в лаборатории натурный испытательный стенд RSE (Random Step Environment), пример которой представлен на рисунке 5.1.1.

RSE представляет собой систему деревянных блоков разной высоты, который при разных конфигурациях могут быть использованы для симуляции различных видов неровностей (ступеней, ям, горок и т.д.). Данная конструкция активно используется для тестирования мобильных роботов в условиях различных препятствий [10, 13, 17, 20].

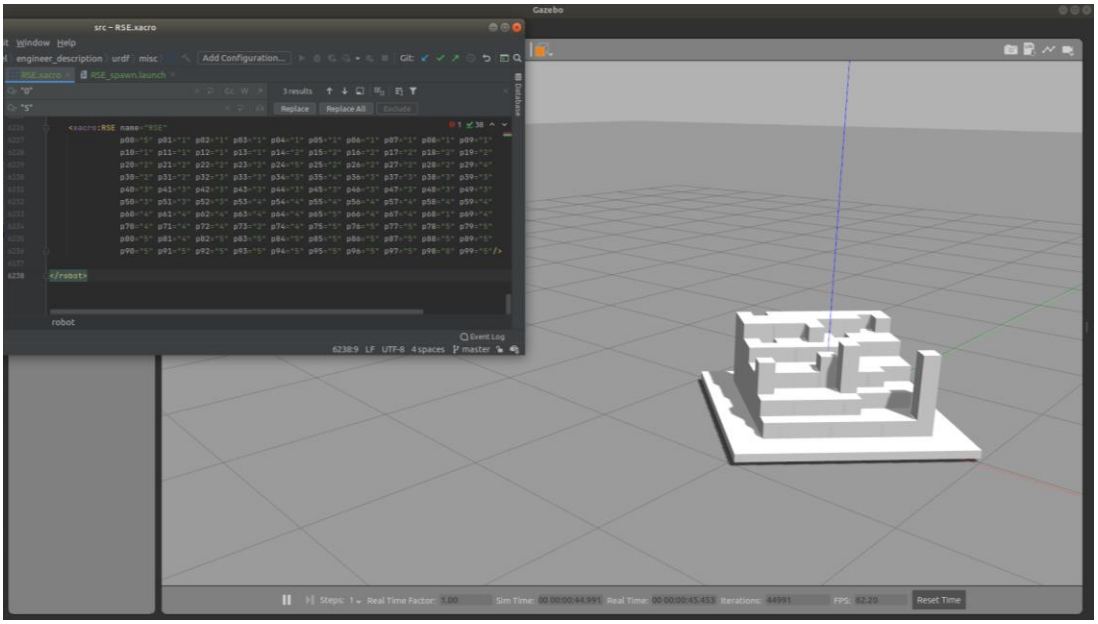

Рисунок 5.1.2. Первый пример вызова функции отстройки RSE в симуляции и результат этого вызова.

Для симуляции описанной конструкции был создан URDF файл с описанием положений брусков и параметрами их высот. Таким образом, в конечном варианте, для добавления в симуляции модели RSE достаточно вызвать написанную функцию и с помощью параметров высоты (от 0 до 5) указать высоты брусков на конкретных местах. При этом, высоты нужно вручную ввести в конфигурационном файле перед вызовом функции построения модели RSE. Пример вызова макроса проиллюстрирован на рисунке 5.1.2.

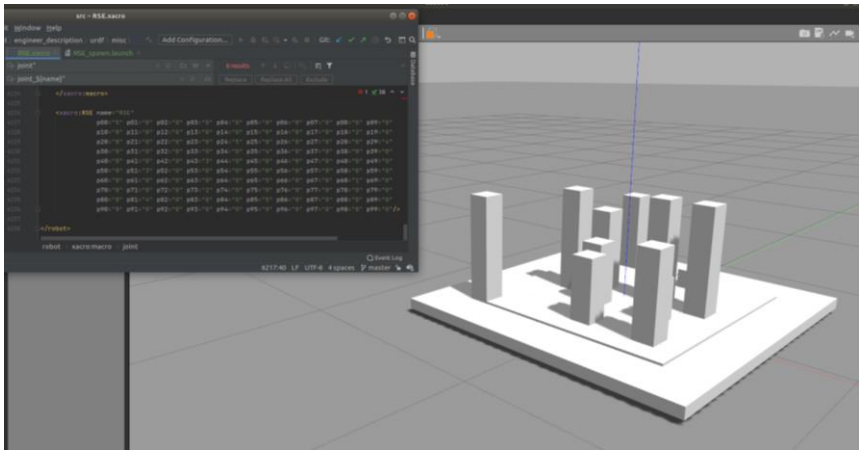

Рисунок 5.1.3. Второй пример вызова функции отстройки RSE в симуляции и результат этого вызова.

Описанная модель состоит из 100 брусков (10 на 10), каждый из которых может быть представлен в 6 различных высотах. На рисунках 5.1.2 и 5.1.3 представлены лишь две конфигурации из множества возможных.

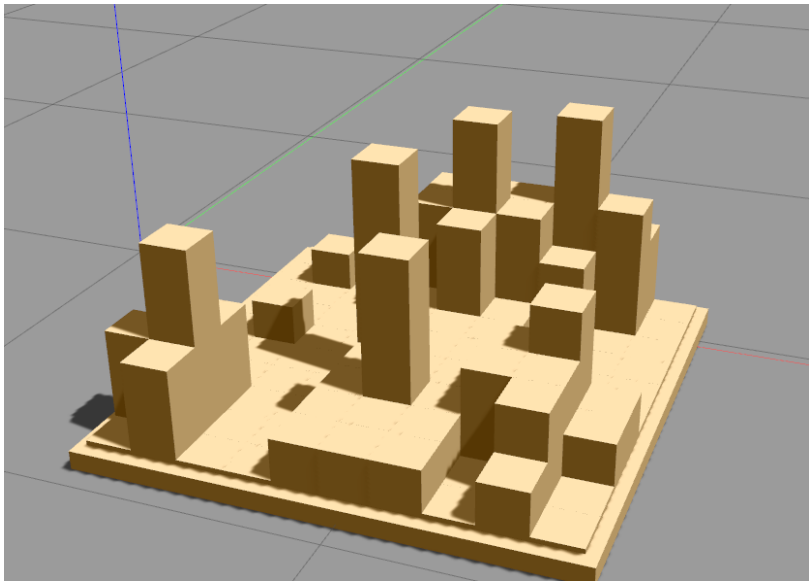

Рисунок 5.1.4. Исправленная симуляционная модель RSE.

В процессе работы над моделью RSE были проведены изменения основания модели, а также произведены небольшие изменения, касающиеся визуального отображения. Модель получила окрас, по тону приближенный к цвету дерева. Рисунок 5.1.4 иллюстрирует одну из конфигураций новой версии модели RSE.

5.2. Эксперименты с движениями по неровным поверхностям

5.2.1. Симуляционная модель в Gazebo

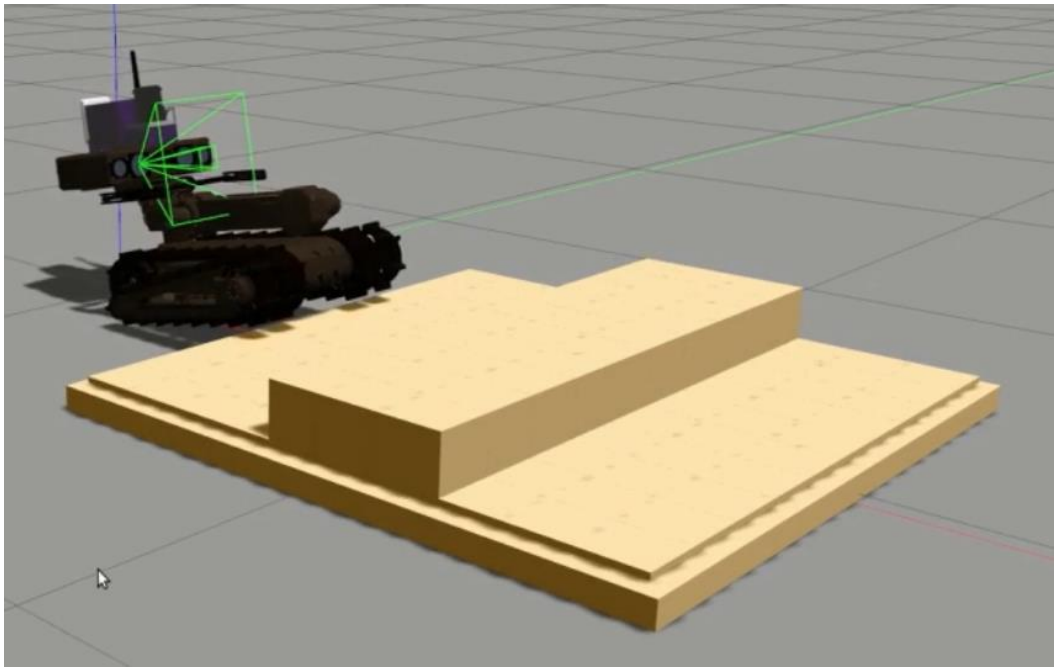

Рисунок 5.2.1.1. Эксперименты в симуляторе Gazebo с моделью RSE.

Когда симуляционная модель Random Step Environment была полностью завершена было решено протестировать ее, проведя несколько экспериментов. Создав конфигурацию простой ступеньки, на RSE была запущена симуляционная модель робота. Рисунок 5.2.1.1 иллюстрирует проведение одного из экспериментов, в процессе которого модели робота необходимо преодолеть представленное препятствие.

В ходе работы было проведено несколько экспериментов, которые в том числе помогли выявить проблемы симуляционной модели RSE – неверно выставленные параметры жесткости и трения. Такая ошибка привела к тому, что при попытке робота начать движение по RSE, ее модель начинала медленно проваливаться сквозь пол. По достижении некоторой глубины подстилающей поверхности моделью RSE с роботом на ней, модель “выпрыгивала” и вставала в нормальное положение, вместе с этим выталкивая модель робота, находящуюся на ней. Проблему удалось решить, определив в URDF модели RSE коэффициенты жесткости и трения для симуляции с тегом <gazebo>. Готовую версию URDF с описанием RSE можно увидеть в приложении Б.

5.2.2. Симуляционная модель в Webots

Помимо модели робота в симуляторе Gazebo была также начата работа над моделью робота «Сервосила Инженера» в симуляторе Webots, где также были проведены несколько экспериментов с движением по неровным поверхностям.

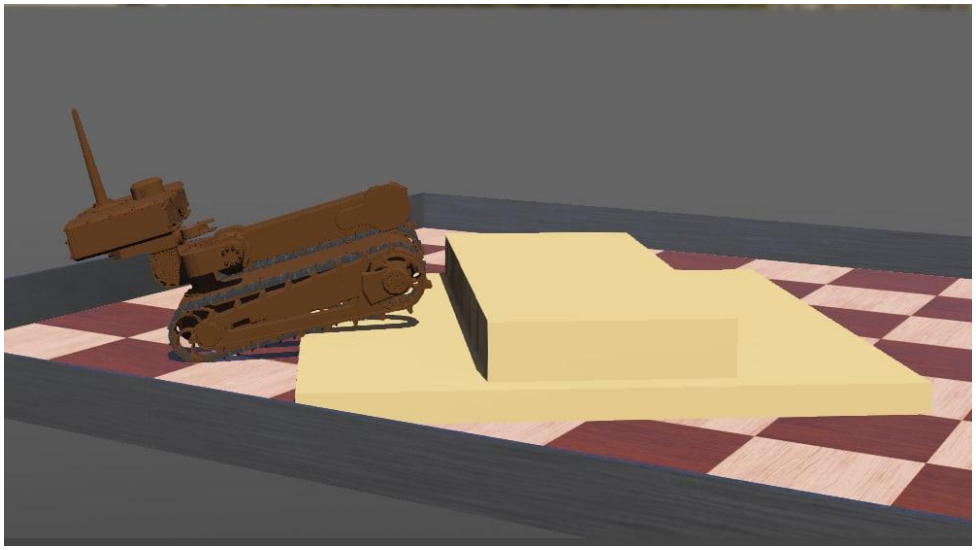

Рисунок 5.2.2.1. Эксперименты с RSE и реальным роботом.

Для проведения эксперимента в симуляционной среде Webots из примитивов была создана модель, повторяющая геометрию RSE в конфигурации с одной ступенькой, которая аналогична модели, использованной в эксперименте в симуляторе Gazebo. На рисунке 5.2.2.1 изображен фрагмент проведенного эксперимента. Симулятор Webots поддерживает симуляцию гусениц, в связи с чем модель робота, созданная в этом симуляторе, не нуждается в симуляционных колесах, что делает модель проще, а саму симуляцию – быстрее. Если в эксперименте с симулятором Gazebo из-за обилия симуляционных колес RTF был снижен до 0.2-0.1, то симуляция в Webots сохраняет RTF на уровне 0.9-1.

5.2.3. Реальный робот

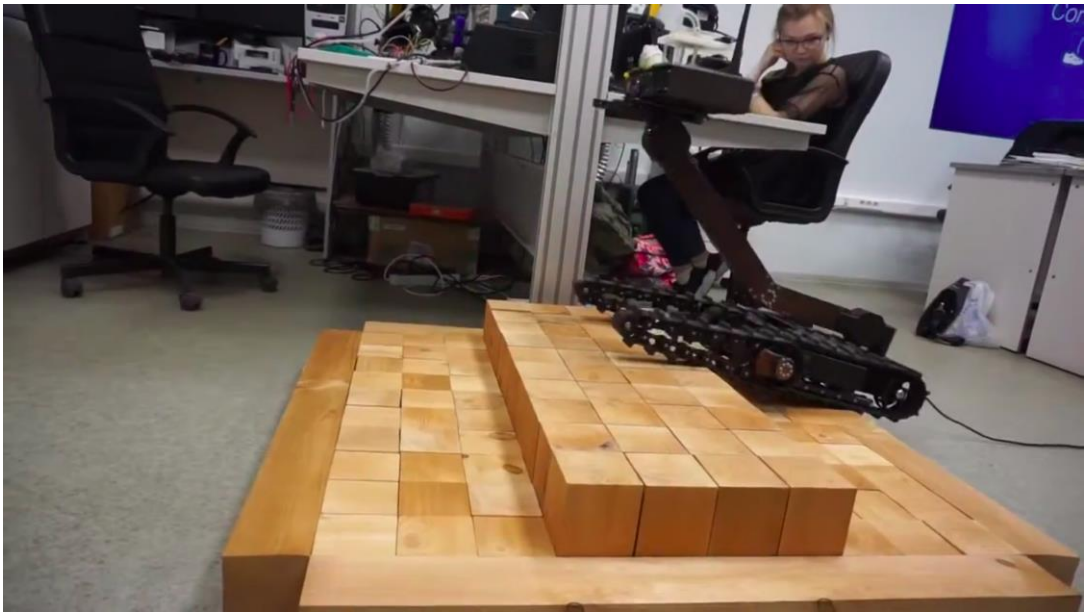

Рисунок 5.2.3.1. Эксперименты с RSE и реальным роботом.

Помимо экспериментов в симуляции, эксперименты с RSE на реальном роботе. Они проводились раньше проведенных исследований [32], но служили ориентиром для поведения модели в симуляции. На рисунке 5.2.3.1 представлен промежуточный этап проведения эксперимента с реальным роботом на RSE в лаборатории. В данном эксперименте, как и в симуляции, модель RSE используется в простой конфигурации, состоящей из одной ступеньки, которую робот успешно преодолевает.

ЗАКЛЮЧЕНИЕ

Создание симуляционной модели для гусеничного робота в симуляторе Gazebo является комплексной задачей. Существует целый ряд трудностей, касающихся симуляции гусениц, так как прямого решения этой проблемы у некоторых симуляторов нет. Это привело к появлению различных подходов к решению проблемы, один из которых рассмотрен и применен в данной работе.

Объектом данной работы стал российский гусеничный робот “Сервосила Инженер” и его симуляционная модель. Отсутствие симуляционной модели для робота, предназначенного для поисково-спасательных операций, делает его применение крайне затруднительным. По этой причине в качестве цели для исследования было предложено создание новой симуляционной модели для робота “Сервосила Инженер”.

Результатом данной работы стало создание улучшенной симуляционной модели робота “Сервосила Инженер”. В рамках работы были реализованы все обозначенные задачи: оптимизированы модели коллизий, улучшена визуальная составляющая модели, добавлены сенсоры и контроллеры, в соответствии с реальным роботом. Был добавлен и настроен стек навигации, который воспроизведен максимально достоверно относительно стека навигации реального робота, а его работа была проверена экспериментально. Также была проведена серия экспериментов с движением модели робота в условиях неровных поверхностей для чего в симуляции была воспроизведена модель RSE (Random Step Environment) и проведено несколько экспериментов по преодолению этого препятствия.

На основании проведенных исследований и разработок было опубликовано 2 статьи [27; 28], сопровождаемые докладами на международных конференциях, таких как Conference on Developments in eSystems Engineering (DeSE 2020) и Siberian Conference on Control and Communications (SIBCON 2021). Также еще одна статья принята к публикации [29], и будет представлена на конференции

Conference on Industrial Engineering and Applications (IEEE ICIEA 2021). Материалы всех вышеупомянутых конференций индексируются в Scopus и Web of Science.

Исходный код данной работы, размещен на Gitlab: https://gitlab.com/LIRS_Projects/Engineer-gazebo-model.

СПИСОК ЛИТЕРАТУРЫ

1. Sokolov M., Lavrenov R., Gabdullin A., Afanasyev I., Magid E. 3D modelling and simulation of a crawler robot in ROS/Gazebo. Proceedings of the 4th International Conference on Control, Mechatronics and Automation. – ACM, 2016. – pp. 61-65.
2. Sokolov, M., Afanasyev, I., Lavrenov, R., Sagitov, A., Sabirova, L., Magid, E. Modelling a crawler-type UGV for urban search and rescue in Gazebo environment. International Conference on Artificial Life and Robotics (ICAROB) – 2017. – pp. 360-362.
3. Moskvin I., Lavrenov R. Modeling Tracks and Controller for Servosila Engineer Robot. Proceedings of 14th International Conference on Electromechanics and Robotics “Zavalishin’s Readings”. – Springer, Singapore, 2020. – pp. 411-422.
4. Pecka M., Zimmermann K., Svoboda T. Fast simulation of vehicles with non-deformable tracks. 2017 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS). – IEEE, 2017. – pp. 6414-6419.
5. Xie S., Shilong B., Bin Z., Huayan P., Jun L., Jason G. The research on obstacle-surmounting capability of six-track robot with four swing arms. IEEE International Conference on Robotics and Biomimetics (ROBIO), – 2013 – pp. 2441-2445.
6. Gerasimov Y., Alishev N., Lavrenov R. Russian mobile robot Servosila Engineer: designing an optimal integration of an extra laser range finder for SLAM purposes. International Conference on Artificial Life and Robotics (ICAROB), – 2018 – pp. 204-207.
7. Kiryanov D., Lavrenov R. Remote Control Application for “Servosila Engineer” on Android Mobile Devices. International Conference on Artificial Life and Robotics (ICAROB), – 2020. – pp. 440-443.

8. Pecka M., Šalanský V., Zimmermann K., Svoboda T. Autonomous flipper control with safety constraints. IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS) – IEEE, 2016 – pp. 2889-2894.
9. Okada Y, Nagatani K, Yoshida K. Semi-autonomous operation of tracked vehicles on rough terrain using autonomous control of active flippers. IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS) – IEEE, 2009 – pp. 2815-2820.
10. Magid E., Tsubouchi T. Static Balance for Rescue Robot Navigation: Translation Motion Discretization Issue within Random Step Environment. Proc. of ICINCO, – 2010, Madeira, Portugal – pp. 415-422.
11. Calisi D., Nardi D., Ohno K., Tadokoro S., A semi-autonomous tracked robot system for rescue missions. SICE Annual Conference, – IEEE, 2008 – pp. 2066-2069.
12. Nagatani K, Yamasaki A, Yoshida K, Yoshida T, Koyanagi E. Semi-autonomous traversal on uneven terrain for a tracked vehicle using autonomous control of active flippers. IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS) – IEEE, 2008 – pp. 2667-2672.
13. Magid E, Tsubouchi T, Koyanagi E, Yoshida T. Static balance for rescue robot navigation: Losing balance on purpose within random step environment. IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS) – IEEE, 2010 – pp. 349-356.
14. Kamimura A, Kurokawa H. High-step climbing by a crawler robot DIR-2-realization of automatic climbing motion. IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS) – IEEE, 2009 – pp. 618-624.
15. Stepson W, Amarasinghe A, Fernando P, Amarasinghe Y. Design and development of a mobile crawling robot with novel halbach array based magnetic wheels. IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS) – IEEE, 2017 – pp. 6561-6566.

16. Inoue T., Shiosawa T., Takagi K. Dynamic Analysis of Motion of Crawler-Type Remotely Operated Vehicles. Journal of Oceanic Engineering, 38(2) – IEEE, 2013 – pp. 375-382.
17. Magid E., Tsubouchi T. Static Balance for Rescue Robot Navigation: Discretizing Rotational Motion within Random Step Environment. Lecture Notes in Artificial Intelligence, 6472, – 2010 – pp. 423-435.
18. Ohno K., Morimura S., Tadokoro S., Koyanagi E., Yoshida T. Semi-autonomous control system of rescue crawler robot having flippers for getting over unknown-steps. In Proceedings of the 2007 IEEE/RSJ International Conference on Intelligent Robots and Systems, – 2007 – pp. 3012–3018.
19. Magid, E., Lavrenov, R., Afanasyev, I. Voronoi-based trajectory optimization for UGV pathplanning. International Conference on Mechanical, System and Control Engineering (ICMSC) – IEEE, 2016 – pp. 383–387.
20. Jacoff, A., Downs, A., Virts, A., Messina, E. Stepfield pallets: repeatable terrain for evaluatingrobot mobility. In Proceedings of the 8th Workshop on Performance Metrics for IntelligentSystems, – 2008 – pp. 29–34.
21. Okada Y. Shared autonomy system for tracked vehicles to traverse rough terrain based on continuous three-dimensional terrain scanning. IEEE/RSJ International Conference On Intelligent Robots and Systems – IEEE, 2010 – pp. 357–362.
22. Xie S. The research on obstacle-surmounting capability of six-track robot with four swingarms. IEEE International Conference on Robotics and Biomimetics (ROBIO) – IEEE, 2013 – pp. 2441–2445.
23. Sheng W., Chen H., Xi N. Navigating a miniature crawler robot for engineered structure inspection. IEEE Trans. Autom. Sci. Eng. 5(2), – IEEE, 2008 – pp. 368–373.
24. Akai N. Autonomous driving based on accurate localization using multilayer LiDAR and dead reckoning. IEEE 20th International Conference on Intelligent TransportationSystems (ITSC), – IEEE, 2017 – pp. 1–6.

25. Mavrin I., Lavrenov R., Svinin M., Sorokin S., and Magid E. Remote control library and GUI development for Russian crawler robot Servosila Engineer. – MATEC Web of Conferences, vol. 161, – EDP Sciences, – 2018 – p. 03016.
26. Michel O. (2004). Cyberbotics Ltd. Webots™: professional mobile robot simulation. International Journal of Advanced Robotic Systems, 1(1), 5.
27. Dobrokvashina A., Lavrenov R., Martinez-Garcia E.A., Bai Y., Improving model of crawler robot Servosila "Engineer" for simulation in ROS/Gazebo. International Conference on Developments in eSystems Engineering (DeSE), - IEEE, 2020
28. Dobrokvashina A., Safin R., Lavrenov R., Bai Y. Improved Graphical User Interface for Crawler Robot Servosila Engineer. Siberian Conference on Control and Communications (SIBCON), - IEEE, 2021
29. Dobrokvashina A., Lavrenov R., Tsoy T., Martinez-Garcia E.A., Bai Y. Navigation stack for the crawler robot Servosila Engineer. International Conference on Industrial Engineering and Applications (ICIEA), - IEEE, 2021
30. Zheng K. Ros navigation tuning guide. arXiv preprint arXiv:1706.09068. 2017.
31. Эксперименты с навигацией робота Сервосила "Инженер" по данным с лазерного дальномера [Электронный ресурс] // YouTube: [сайт], URL: https://youtu.be/PwwKyJuyxGo (дата обращения: 29.05.2021).
32. Random Step Environment Unboxing [Электронный ресурс] // YouTube: [сайт], URL: https://www.youtube.com/watch?v=lt5fEpQC028 (дата обращения: 29.05.2021).
33. Официальный сайт компании «Сервосила» [Электронный ресурс] // Сервосила: [сайт], URL: https://www.servosila.com/en/index.shtml (дата обращения: 29.05.2021).
34. Blackburn M., Everett H., Laird R. After action report to the joint program office: Center the robotic assisted search and rescue (CRASAR) related efforts at the world trade center (No. SSC/SD-TD-3141), - Space And Naval Warfare Systems Center San Diego CA, 2002.

35. Nagatani K. et al. Emergency response to the nuclear accident at the Fukushima Daiichi Nuclear Power Plants using mobile rescue robots. Journal of Field Robotics 30.1 (2013), pp. 44-63.

ПРИЛОЖЕНИЯ

Приложение А

Описание модели робота “Сервосила Инженер”

A1. Фрагмент UDRF файла с описанием манипулятора робота “Сервосила Инженер” - engineer_4dof_arm.xacro

```
<!-- head -->
  <link name="head_link">
   <visual>
    <geometry>
     <mesh filename="package://engineer_description/meshes/scaled_head.dae" scale="0.1 0.1 0.1"/>
    </geometry>
   </visual>
   <collision>
    <geometry>
     <mesh filename="package://engineer_description/meshes/collision/scaled_Head.dae" scale="0.1 0.1 0.1"/>
    </geometry>
   </collision>
   <xacro:cuboid_inertia mass="${head_mass}" length="0.27" height="0.06" width="0.2">
    <origin xyz="0.0272 0.09 0.0225" rpy="0 0 0"/>
   </xacro:cuboid_inertia>
  </link>

  <joint name="head" type="revolute">
   <parent link="elbow_link"/>
   <child link="head_link"/>
   <origin xyz="0.01 0 -0.42" rpy="-${pi} 0 0"/>
   <axis xyz="1 0 0"/>
   <dynamics friction="${friction}" damping="${damping}"/>
   <limit lower="${head_llimit}" upper="${head_ulimit}" effort="${head_mass * 50}" velocity="${joints_vlimit}"/>
  </joint>

  <gazebo reference="head_link">
   <selfCollide>false</selfCollide>
   <kp>${kp}</kp>
   <kd>${kd}</kd>
  </gazebo>

  <transmission name="elbow_head_trans">
   <type>transmission_interface/SimpleTransmission</type>
   <actuator name="elbow_head_motor">
```

```
            <hardwareInterface>hardware_interface/EffortJointInterface</hardwareInterface>
            <mechanicalReduction>1</mechanicalReduction>
          </actuator>
          <joint name="head">
            <hardwareInterface>hardware_interface/EffortJointInterface</hardwareInterface>
          </joint>
        </transmission>

        <link name="laser_base1_link">
          <visual>
            <geometry>
              <mesh       filename="package://engineer_description/meshes/laser_scan_base1_2.dae"
scale="0.25 0.25 0.25"/>
            </geometry>
          </visual>
          <collision>
            <geometry>
       <mesh filename="package://engineer_description/meshes/collision/laser_scan_base1_2.dae"
scale="0.25 0.25 0.25"/>
            </geometry>
          </collision>
          <inertial>
            <mass value="0.1" />
            <inertia ixx="0.0001" ixy="0.0" ixz="0.0" iyy="0.0001" iyz="0.0" izz="0.0001" />
          </inertial>
        </link>

        <joint name="laser1" type="fixed">
          <parent link="head_link"/>
          <child link="laser_base1_link"/>
          <origin xyz="0.02 0.14 -0.01" rpy="-${pi/2} ${pi} 0"/>
        </joint>

        <gazebo reference="laser_base1_link">
          <selfCollide>false</selfCollide>
          <kp>${kp}</kp>
          <kd>${kd}</kd>
        </gazebo>

        <link name="laser_base2_link">
          <visual>
            <geometry>
              <mesh          filename="package://engineer_description/meshes/laser_scan_base2.dae"
scale="0.25 0.25 0.25"/>
            </geometry>
          </visual>
          <collision>
            <geometry>
         <mesh  filename="package://engineer_description/meshes/collision/laser_scan_base2.dae"
scale="0.25 0.25 0.25"/>
```

```
      </geometry>
    </collision>
    <inertial>
      <mass value="0.05" />
      <inertia ixx="0.0001" ixy="0.0" ixz="0.0" iyy="0.0001" iyz="0.0" izz="0.0001" />
    </inertial>
  </link>

  <joint name="laser2" type="revolute">
    <parent link="laser_base1_link"/>
    <child link="laser_base2_link"/>
    <origin xyz="0.005 -0.04 0.041" rpy="0 0 0"/>
    <axis xyz="1 0 0"/>
    <dynamics friction="1.0" damping="1.0"/>
    <limit lower="0" upper="1.3" effort="10" velocity="${joints_vlimit}"/>
  </joint>

  <gazebo reference="laser_base2_link">
    <selfCollide>false</selfCollide>
    <kp>${kp}</kp>
    <kd>${kd}</kd>
  </gazebo>

  <transmission name="head_laser_trans">
    <type>transmission_interface/SimpleTransmission</type>
    <actuator name="head_laser_motor">
      <hardwareInterface>hardware_interface/EffortJointInterface</hardwareInterface>
      <mechanicalReduction>1</mechanicalReduction>
    </actuator>
    <joint name="laser2">
      <hardwareInterface>hardware_interface/EffortJointInterface</hardwareInterface>
    </joint>
  </transmission>

  <!-- Hokuyo Laser -->
  <link name="hokuyo_link">
    <collision>
      <origin xyz="0 0 0" rpy="0 0 0"/>
      <geometry>
    <mesh filename="package://engineer_description/meshes/collision/laser_scan.dae"
scale="0.25 0.25 0.25"/>
      </geometry>
    </collision>

    <visual>
      <origin xyz="0 0 0" rpy="0 0 0"/>
      <geometry>
        <mesh filename="package://engineer_description/meshes/laser_scan.dae" scale="0.25
0.25 0.25"/>
      </geometry>
```

```
  </visual>

  <inertial>
    <mass value="0.1" />
    <inertia ixx="0.001" ixy="0.0" ixz="0.0" iyy="0.001" iyz="0.0" izz="0.001" />
  </inertial>
</link>

<joint name="hokuyo_joint" type="fixed">
  <axis xyz="0 1 0" />
  <origin xyz="0 0.03 0.005" rpy="0 0 -${pi/2}"/>
  <parent link="laser_base2_link"/>
  <child link="hokuyo_link"/>
</joint>

<xacro:macro name="head_camera" params ="number width height fps *origin">
  <joint name="camera${number}_joint" type="fixed">
    <axis xyz="0 1 0" />
    <xacro:insert_block name="origin"/>
    <parent link="head_link"/>
    <child link="camera${number}_link"/>
  </joint>

  <!-- Camera -->
  <link name="camera${number}_link">
    <collision>
      <origin xyz="0 0 0" rpy="0 0 0"/>
      <geometry>
        <box size="0.0001 0.0001 0.0001"/>
      </geometry>
    </collision>

    <visual>
      <origin xyz="0 0 0" rpy="0 0 0"/>
      <geometry>
        <box size="0.0001 0.0001 0.0001"/>
      </geometry>
    </visual>

    <xacro:cuboid_inertia mass="0.1" length="0.01" width="0.01" height="0.01">
      <origin xyz="0 0 0" rpy="0 0 0"/>
    </xacro:cuboid_inertia>
  </link>


  <joint name="camera${number}_optical_joint" type="fixed">
    <origin xyz="0 0 0" rpy="${-pi/2} 0 ${-pi/2}"/>
    <parent link="camera${number}_link"/>
    <child link="camera${number}_link_optical"/>
  </joint>
```

```
  <link name="camera${number}_link_optical">
  </link>

  <gazebo reference="camera${number}_link">
    <sensor type="camera" name="camera_${number}">
      <update_rate>${fps}</update_rate>
      <camera name="head_${number}">
        <horizontal_fov>1.3962634</horizontal_fov>
        <image>
          <width>${width}</width>
          <height>${height}</height>
          <format>R8G8B8</format>
        </image>
        <clip>
          <near>0.02</near>
          <far>300</far>
        </clip>
        <noise>
          <type>gaussian</type>
          <mean>0.0</mean>
          <stddev>0.007</stddev>
        </noise>
      </camera>
      <plugin name="camera${number}_controller" filename="libgazebo_ros_camera.so">
        <alwaysOn>true</alwaysOn>
        <updateRate>0.0</updateRate>
        <cameraName>camera${number}</cameraName>
        <imageTopicName>image${number}_raw</imageTopicName>
        <cameraInfoTopicName>camera_info</cameraInfoTopicName>
        <frameName>camera${number}_link_optical</frameName>]
        <hackBaseline>0.0</hackBaseline>
        <distortionK1>0.0</distortionK1>
        <distortionK2>0.0</distortionK2>
        <distortionK3>0.0</distortionK3>
        <distortionT1>0.0</distortionT1>
        <distortionT2>0.0</distortionT2>
        <CxPrime>0</CxPrime>
        <Cx>0.0</Cx>
        <Cy>0.0</Cy>
        <focalLength>0.0</focalLength>
      </plugin>
    </sensor>
  </gazebo>
</xacro:macro>

<xacro:head_camera number="1" width="1280" height="720" fps="50.0">
  <origin xyz="-0.035 0.11 -0.04" rpy="-${pi/2} ${pi/2} 0"/>
</xacro:head_camera>
```

```
        <xacro:head_camera number="2" width="640" height="480" fps="30.0">
          <origin xyz="0.13 0.11 -0.04" rpy="-${pi/2} ${pi/2} 0"/>
        </xacro:head_camera>

        <xacro:head_camera number="3" width="640" height="480" fps="30.0">
          <origin xyz="-0.085 0.11 -0.04" rpy="-${pi/2} ${pi/2} 0"/>
        </xacro:head_camera>

        <xacro:head_camera number="4" width="640" height="480" fps="30.0">
          <origin xyz="-0.035 0.11 0.135" rpy="-${pi/2} -${pi/2} 0"/>
        </xacro:head_camera>

        <!-- hokuyo -->
        <gazebo reference="hokuyo_link">
          <sensor type="gpu_ray" name="head_hokuyo_sensor">
            <pose>0 0 0 0 0 0</pose>
            <visualize>false</visualize>
            <update_rate>40</update_rate>
            <ray>
              <scan>
                <horizontal>
                  <samples>720</samples>
                  <resolution>1</resolution>
                  <min_angle>-1.570796</min_angle>
                  <max_angle>1.570796</max_angle>
                </horizontal>
              </scan>
              <range>
                <min>0.10</min>
                <max>30.0</max>
                <resolution>0.01</resolution>
              </range>
              <noise>
                <type>gaussian</type>
                <!-- Noise parameters based on published spec for Hokuyo laser
                     achieving "+-30mm" accuracy at range < 10m.  A mean of 0.0m and
                     stddev of 0.01m will put 99.7% of samples within 0.03m of the true
                     reading. -->
                <mean>0.0</mean>
                <stddev>0.01</stddev>
              </noise>
            </ray>
      <plugin                                          name="gazebo_ros_head_hokuyo_controller"
filename="libgazebo_ros_gpu_laser.so">
              <topicName>laser_scan</topicName>
              <frameName>hokuyo_link</frameName>
            </plugin>
          </sensor>
        </gazebo>
```

```
<joint name="imu_joint" type="fixed">
  <axis xyz="0 1 0" />
  <origin xyz="0.02 0.14 0.08" rpy="-${pi/2} ${pi/2} 0"/>
  <parent link="head_link"/>
  <child link="imu_link"/>
</joint>

<!-- IMU -->
<link name="imu_link">
  <collision>
    <origin xyz="0 0 0" rpy="0 0 0"/>
    <geometry>
      <box size="0.01 0.01 0.01"/>
    </geometry>
  </collision>

  <visual>
    <origin xyz="0 0 0" rpy="0 0 0"/>
    <geometry>
      <box size="0.01 0.01 0.01"/>
    </geometry>
  </visual>

  <xacro:cuboid_inertia mass="0.1" length="0.01" width="0.01" height="0.01">
    <origin xyz="0 0 0" rpy="0 0 0"/>
  </xacro:cuboid_inertia>
</link>

<gazebo reference="imu_link">
  <gravity>true</gravity>
  <sensor name="imu_sensor" type="imu">
    <always_on>true</always_on>
    <update_rate>100</update_rate>
    <visualize>false</visualize>
    <topic>__default_topic__</topic>
    <plugin filename="libgazebo_ros_imu_sensor.so" name="imu_plugin">
      <topicName>/imu</topicName>
      <bodyName>imu_link</bodyName>
      <updateRateHZ>10.0</updateRateHZ>
      <gaussianNoise>0.0</gaussianNoise>
      <xyzOffset>0 0 0</xyzOffset>
      <rpyOffset>0 0 0</rpyOffset>
      <frameName>engineer/imu_link</frameName>
      <initialOrientationAsReference>false</initialOrientationAsReference>
    </plugin>
    <pose>0 0 0 0 0 0</pose>
  </sensor>
</gazebo>
```

```
<gazebo reference="head_link">
  <light type="spot" name="spot">
    <pose>0.08 0.11 -0.04 0 0 0</pose>
    <diffuse>1 1 1 1</diffuse>
    <specular>.2 .2 .2 1</specular>
    <attenuation>
      <range>10</range>
      <linear>0.01</linear>
      <constant>0.2</constant>
      <quadratic>0.0</quadratic>
    </attenuation>
    <direction>0 0 -1</direction>
    <spot>
      <inner_angle>0.1</inner_angle>
      <outer_angle>0.5</outer_angle>
      <falloff>1.2</falloff>
    </spot>
    <cast_shadows>true</cast_shadows>
  </light>
</gazebo>
```

А2. Фрагмент UDRF файла с описанием симуляционных колес робота “Сервосила Инженер” - diff_drive.xacro

```
<xacro:macro name="wheel"
             params="name_link x y z roll pitch yaw radius wheel_length mass parent">
    <link name="${name_link}">
      <visual>
        <origin xyz="0 0 0" rpy="0 0 0"/>
        <geometry>
 <cylinder radius='${radius*scale*inv_scale}' length="${wheel_length*scale*inv_scale}"/>
         </geometry>
      </visual>
      <collision>
        <origin xyz="0 0 0" rpy="0 0 0"/>
        <geometry>
          <cylinder radius='${radius*scale}' length="${wheel_length*scale}"/>
        </geometry>
      </collision>
      <xacro:cylinder_inertia radius="0.001" height="0.001" mass="${mass}">
        <origin xyz="0 0 0" rpy="0 0 0"/>
      </xacro:cylinder_inertia>

    </link>
    <joint name="joint_${name_link}" type="continuous">
      <parent link="${parent}"/>
      <child link="${name_link}"/>
      <origin xyz="${x} ${y} ${z}" rpy="${roll} ${pitch} ${yaw}"/>
      <axis xyz="0 0 1"/>
      <limit effort="100.0" lower="0.0" upper="0.0" velocity="10.0"/>
    </joint>

    <gazebo reference="${name_link}">
      <selfCollide>false</selfCollide>
      <mu1>100</mu1>
      <mu2>200</mu2>
      <kp>1e6</kp>
      <kd>100</kd>
    </gazebo>

    <transmission name="tran_joint_${name_link}">
      <type>transmission_interface/SimpleTransmission</type>
      <joint name="joint_${name_link}">
        <hardwareInterface>hardware_interface/VelocityJointInterface</hardwareInterface>
      </joint>
      <actuator name="motor_joint_${name_link}">
        <mechanicalReduction>1</mechanicalReduction>
      </actuator>
    </transmission>
  </xacro:macro>
```

```
    <xacro:wheel name_link="right1_base_link_wheel" x="${1.86*scale}" y="${-1.2*scale}"
z="${-0.95*scale}" roll="0" pitch="${PI*0.5}" yaw="${PI*0.5}"
                 radius="${big_wheel_radius}"                wheel_length="${wheel_length}"
mass="${wheel_mass}" parent="base_link"/>
      <xacro:wheel name_link="left1_base_link_wheel" x="${1.86*scale}" y="${1.2*scale}"
z="${-0.95*scale}" roll="0" pitch="${PI*0.5}" yaw="${PI*0.5}"
                 radius="${big_wheel_radius}"                wheel_length="${wheel_length}"
mass="${wheel_mass}" parent="base_link"/>

    <xacro:wheel name_link="right2_base_link_wheel" x="${1.6533*scale}" y="${-1.2*scale}"
z="${-0.95*scale}" roll="0" pitch="${PI*0.5}" yaw="${PI*0.5}"
                 radius="${big_wheel_radius}"                wheel_length="${wheel_length}"
mass="${wheel_mass}" parent="base_link"/>
      <xacro:wheel name_link="left2_base_link_wheel" x="${1.6533*scale}" y="${1.2*scale}"
z="${-0.95*scale}" roll="0" pitch="${PI*0.5}" yaw="${PI*0.5}"
                 radius="${big_wheel_radius}"                wheel_length="${wheel_length}"
mass="${wheel_mass}" parent="base_link"/>

    <xacro:wheel name_link="right3_base_link_wheel" x="${1.4466*scale}" y="${-1.2*scale}"
z="${-0.95*scale}" roll="0" pitch="${PI*0.5}" yaw="${PI*0.5}"
                 radius="${big_wheel_radius}"                wheel_length="${wheel_length}"
mass="${wheel_mass}" parent="base_link"/>
      <xacro:wheel name_link="left3_base_link_wheel" x="${1.4466*scale}" y="${1.2*scale}"
z="${-0.95*scale}" roll="0" pitch="${PI*0.5}" yaw="${PI*0.5}"
                 radius="${big_wheel_radius}"                wheel_length="${wheel_length}"
mass="${wheel_mass}" parent="base_link"/>

    <xacro:wheel name_link="right4_base_link_wheel" x="${1.2399*scale}" y="${-1.2*scale}"
z="${-0.95*scale}" roll="0" pitch="${PI*0.5}" yaw="${PI*0.5}"
                 radius="${big_wheel_radius}"                wheel_length="${wheel_length}"
mass="${wheel_mass}" parent="base_link"/>
      <xacro:wheel name_link="left4_base_link_wheel" x="${1.2399*scale}" y="${1.2*scale}"
z="${-0.95*scale}" roll="0" pitch="${PI*0.5}" yaw="${PI*0.5}"
                 radius="${big_wheel_radius}"                wheel_length="${wheel_length}"
mass="${wheel_mass}" parent="base_link"/>

    <xacro:wheel name_link="right5_base_link_wheel" x="${1.0332*scale}" y="${-1.2*scale}"
z="${-0.95*scale}" roll="0" pitch="${PI*0.5}" yaw="${PI*0.5}"
                 radius="${big_wheel_radius}"                wheel_length="${wheel_length}"
mass="${wheel_mass}" parent="base_link"/>
      <xacro:wheel name_link="left5_base_link_wheel" x="${1.0332*scale}" y="${1.2*scale}"
z="${-0.95*scale}" roll="0" pitch="${PI*0.5}" yaw="${PI*0.5}"
                 radius="${big_wheel_radius}"                wheel_length="${wheel_length}"
mass="${wheel_mass}" parent="base_link"/>
```

А3. Фрагмент UDRF файла с захвата робота “Сервосила Инженер” - engineer_ee.xacro

```
<xacro:macro name="ee_part" params="defen parent reflect">

  <link name="part_one_${defen}">
    <visual>
      <geometry>
    <mesh  filename="package://engineer_description/meshes/scaled_ee_1link_${defen}.dae" scale="0.1 0.1 0.1"/>
      </geometry>
      <material name="brown"/>
    </visual>
    <collision>
      <geometry>
<mesh filename="package://engineer_description/meshes/collision/scaled_${defen}_ee_1.dae"  scale="0.1 0.1 0.1"/>
      </geometry>
    </collision>
    <xacro:cylinder_inertia mass="0.2" height="0.05" radius="0.05">
        <origin xyz="0 0 0" rpy="${pi} 0 0" />
    </xacro:cylinder_inertia>
  </link>

<gazebo reference="part_one_${defen}">
  <selfCollide>false</selfCollide>
  <mu1>200</mu1>
  <mu2>100</mu2>
  <kp>10000000</kp>
  <kd>1</kd>
</gazebo>

  <joint name="${parent}_p1_${defen}_ee" type="revolute">
    <parent link="${parent}"/>
    <child link="part_one_${defen}"/>
    <dynamics friction="1.0" damping="1.0"/>
    <axis xyz="0 1 0"/>
    <xacro:if value="${reflect == -1}">
    <mimic joint="${parent}_p1_left_ee" multiplier="-1.0"/>
    </xacro:if>
    <limit lower="-1.45" upper="0.6" effort="0.5" velocity="${joints_vlimit}"/>
    <origin xyz="${reflect*0.016+0.022} 0.057 -0.058" rpy="${pi} ${-reflect*pi/2} 0"/>
  </joint>

<xacro:if value="${reflect == 1}">
<transmission name="tran_${parent}_p1_${defen}_ee">
    <type>transmission_interface/SimpleTransmission</type>
```

```
            <joint name="${parent}_p1_${defen}_ee">
              <hardwareInterface>hardware_interface/EffortJointInterface</hardwareInterface>
            </joint>
            <actuator name="motor_${parent}_p1_${defen}_ee">
              <hardwareInterface>hardware_interface/EffortJointInterface</hardwareInterface>
              <mechanicalReduction>1</mechanicalReduction>
            </actuator>
          </transmission>
        </xacro:if>

        <xacro:if value="${reflect == -1}">
          <gazebo>
            <plugin                                name="${parent}_p1_${defen}_ee_mimic_joint_plugin"
filename="libroboticsgroup_gazebo_mimic_joint_plugin.so">
            <joint>${parent}_p1_${defen}_ee</joint>
            <mimicJoint>${parent}_p1_left_ee</mimicJoint>
            <multiplier>1.0</multiplier>
            <maxEffort>1.0</maxEffort>

            <robotNamespace>engineer</robotNamespace>
          </plugin>
        </gazebo>
        </xacro:if>

        <link name="part_two_${defen}">
          <visual>
            <geometry>
           <mesh filename="package://engineer_description/meshes/scaled_ee_2link_${defen}.dae"
scale="0.1 0.1 0.1"/>
            </geometry>
            <material name="brown"/>
          </visual>
          <collision>
            <geometry>
        <mesh
filename="package://engineer_description/meshes/collision/scaled_${defen}_ee_2.dae"  scale="0.1
0.1 0.1"/>
            </geometry>
          </collision>
          <xacro:cylinder_inertia mass="0.2" height="0.05" radius="0.05">
                <origin xyz="0 0 0" rpy="${pi} 0 0" />
          </xacro:cylinder_inertia>
        </link>

        <gazebo reference="part_two_${defen}">
          <selfCollide>false</selfCollide>
          <mu1>200</mu1>
          <mu2>100</mu2>
          <kp>10000000</kp>
          <kd>1</kd>
```

```
    </gazebo>

    <joint name="${parent}_p2_${defen}_ee" type="continuous">
       <parent link="${parent}"/>
       <child link="part_two_${defen}"/>
       <dynamics friction="1.0" damping="1.0"/>
       <axis xyz="0 1 0"/>
       <mimic joint="${parent}_p1_${defen}_ee" multiplier="1.0"/>
       <origin xyz="${reflect*0.008+0.022} 0.057 -0.095" rpy="${pi} ${-reflect*pi/2} 0"/>
    </joint>

    <gazebo>
      <plugin                                  name="${parent}_p2_${defen}_ee_mimic_joint_plugin"
filename="libroboticsgroup_gazebo_mimic_joint_plugin.so">
       <joint>${parent}_p2_${defen}_ee</joint>
       <mimicJoint>${parent}_p1_${defen}_ee</mimicJoint>
       <multiplier>1.0</multiplier>
       <maxEffort>1.0</maxEffort>

       <robotNamespace>engineer</robotNamespace>
      </plugin>
     </gazebo>

    <link name="part_three_${defen}">
       <visual>
          <geometry>
        <mesh  filename="package://engineer_description/meshes/scaled_ee_3link_${defen}.dae"
scale="0.1 0.1 0.1"/>
          </geometry>
          <material name="brown"/>
       </visual>
       <collision>
          <geometry>
    <mesh
filename="package://engineer_description/meshes/collision/scaled_${defen}_ee_3.dae"  scale="0.1
0.1 0.1"/>
          </geometry>
       </collision>
       <xacro:cylinder_inertia mass="0.1" height="0.05" radius="0.05">
              <origin xyz="0 0 0" rpy="${pi/2} 0 0" />
       </xacro:cylinder_inertia>
    </link>

    <gazebo reference="part_three_${defen}">
      <selfCollide>false</selfCollide>
      <mu1>200</mu1>
      <mu2>100</mu2>
      <kp>10000000</kp>
      <kd>1</kd>
    </gazebo>
```

```
    <joint name="${parent}_p3_${defen}_ee" type="continuous">
       <parent link="part_one_${defen}"/>
       <child link="part_three_${defen}"/>
       <dynamics friction="1.0" damping="1.0"/>
       <axis xyz="0 1 0"/>
       <mimic joint="${parent}_p1_${defen}_ee" multiplier="-1"/>
       <origin xyz="${reflect*0.008} 0 0.145" rpy="0 ${-reflect*pi/2-reflect*pi/10} 0"/>
    </joint>

    <gazebo>
    <plugin                                    name="${parent}_p3_${defen}_ee_mimic_joint_plugin"
filename="libroboticsgroup_gazebo_mimic_joint_plugin.so">
        <joint>${parent}_p3_${defen}_ee</joint>
        <mimicJoint>${parent}_p1_${defen}_ee</mimicJoint>
        <multiplier>-1.0</multiplier>
        <maxEffort>1.0</maxEffort>

        <robotNamespace>engineer</robotNamespace>
      </plugin>
     </gazebo>

     </xacro:macro>

     <xacro:macro name="engineer_end_effector" params="parent">

       <xacro:ee_part defen="left" parent="${parent}" reflect="1"/>
       <xacro:ee_part defen="right" parent="${parent}" reflect="-1"/>

     </xacro:macro>
```

А4. Фрагмент UDRF файла сописанием базы робота “Сервосила Инженер” - engineer_mobile_base.xacro

```
<xacro:macro name="engineer_base" params="footprint_offset">
  <link name="base_footprint">
   <visual>
    <geometry>
     <box size="0.001 0.001 0.001"/>
    </geometry>
   </visual>
  </link>

  <gazebo reference="base_footprint">
   <turnGravityOff>false</turnGravityOff>
  </gazebo>

  <joint name="base_footprint_joint" type="fixed">
   <origin xyz="0 0 ${footprint_offset}" rpy="0 0 0"/>
   <parent link="base_footprint"/>
   <child link="base_link"/>
   <dynamics friction="1.0" damping="1.0"/>
  </joint>

  <link name="base_link">
   <visual>
    <geometry>
     <mesh filename="package://engineer_description/meshes/scaled_base_nt.dae"
scale="0.1 0.1 0.1"/>
    </geometry>
   </visual>
   <collision>
    <geometry>
     <mesh filename="package://engineer_description/meshes/collision/scaled_Base_nt.dae"
scale="0.1 0.1 0.1"/>
    </geometry>
   </collision>
   <xacro:cuboid_inertia mass="${base_mass}" length="0.45" width="0.15" height="0.1">
    <origin xyz="0 0 -0.015" rpy="1.57 0 1.57"/>
   </xacro:cuboid_inertia>
  </link>

  <gazebo reference="base_link">
   <selfCollide>false</selfCollide>
   <mu1>100</mu1>
   <mu2>200</mu2>
   <kp>1e6</kp>
   <kd>1</kd>
  </gazebo>
 </xacro:macro>
```

Приложение Б

Фрагмент UDRF файла с описанием симуляционной модели Random Step Environment - RSE.xacro

```
  <xacro:property name="height0" value="0.01"/>
  <xacro:property name="height1" value="0.1"/>
  <xacro:property name="height2" value="0.2"/>
  <xacro:property name="height3" value="0.3"/>
  <xacro:property name="height4" value="0.4"/>
  <xacro:property name="height5" value="0.5"/>

  <xacro:macro name="brown" params="link_name">
    <gazebo reference="${link_name}">
      <visual>
        <material>
          <ambient>0.8 0.5 0.0 1.0</ambient>
        </material>
      </visual>
      <kp>10000000</kp>
      <kd>1</kd>
    </gazebo>
  </xacro:macro>

<xacro:macro name="RSE" params="name
  p00 p01 p02 p03 p04 p05 p06 p07 p08 p09
  p10 p11 p12 p13 p14 p15 p16 p17 p18 p19
  p20 p21 p22 p23 p24 p25 p26 p27 p28 p29
  p30 p31 p32 p33 p34 p35 p36 p37 p38 p39
  p40 p41 p42 p43 p44 p45 p46 p47 p48 p49
  p50 p51 p52 p53 p54 p55 p56 p57 p58 p59
  p60 p61 p62 p63 p64 p65 p66 p67 p68 p69
  p70 p71 p72 p73 p74 p75 p76 p77 p78 p79
  p80 p81 p82 p83 p84 p85 p86 p87 p88 p89
  p90 p91 p92 p93 p94 p95 p96 p97 p98 p99">

    <!-- Tag on the external object -->
    <link name="${name}_base">
      <collision>
        <geometry>
          <box size="1.05 1.05 0.05"/>
        </geometry>
      </collision>
      <visual>
        <geometry>
          <box size="1.05 1.05 0.04"/>
        </geometry>
```

```
            </visual>
            <xacro:default_inertia mass="5"/>
        </link>

        <xacro:brown link_name="${name}_base"/>
    <!--    <xacro:loop links_qty="1" row="0" col="0"/>-->
        <link name="p00_block_${name}">
            <collision>
                <xacro:if value="${p00 != 1 and p00 != 2 and p00 != 3 and p00 != 4 and p00 !=
5}">
                    <geometry><box size="0.1 ${height0} 0.1"/></geometry>
                    <origin xyz="0 ${height0/2+0.025} 0" rpy="0 0 0"/>
                </xacro:if>
                <xacro:if value="${p00 == 1}">
                    <geometry><box size="0.1 ${height1} 0.1"/></geometry>
                    <origin xyz="0 ${height1/2+0.025} 0" rpy="0 0 0"/>
                </xacro:if>
                <xacro:if value="${p00 == 2}">
                    <geometry><box size="0.1 ${height2} 0.1"/></geometry>
                    <origin xyz="0 ${height2/2+0.025} 0" rpy="0 0 0"/>
                </xacro:if>
                <xacro:if value="${p00 == 3}">
                    <geometry><box size="0.1 ${height3} 0.1"/></geometry>
                    <origin xyz="0 ${height3/2+0.025} 0" rpy="0 0 0"/>
                </xacro:if>
                <xacro:if value="${p00 == 4}">
                    <geometry><box size="0.1 ${height4} 0.1"/></geometry>
                    <origin xyz="0 ${height4/2+0.025} 0" rpy="0 0 0"/>
                </xacro:if>
                <xacro:if value="${p00 == 5}">
                    <geometry><box size="0.1 ${height5} 0.1"/></geometry>
                    <origin xyz="0 ${height5/2+0.025} 0" rpy="0 0 0"/>
                </xacro:if>
            </collision>
            <visual>
                <xacro:if value="${p00 != 1 and p00 != 2 and p00 != 3 and p00 != 4 and p00 !=
5}">
                    <geometry><box size="0.1 ${height0} 0.1"/></geometry>
                    <origin xyz="0 ${height0/2+0.025} 0" rpy="0 0 0"/>
                </xacro:if>
                <xacro:if value="${p00 == 1}">
                    <geometry><box size="0.1 ${height1} 0.1"/></geometry>
                    <origin xyz="0 ${height1/2+0.025} 0" rpy="0 0 0"/>
                </xacro:if>
                <xacro:if value="${p00 == 2}">
                    <geometry><box size="0.1 ${height2} 0.1"/></geometry>
                    <origin xyz="0 ${height2/2+0.025} 0" rpy="0 0 0"/>
                </xacro:if>
                <xacro:if value="${p00 == 3}">
                    <geometry><box size="0.1 ${height3} 0.1"/></geometry>
```

```
            <origin xyz="0 ${height3/2+0.025} 0" rpy="0 0 0"/>
          </xacro:if>
          <xacro:if value="${p00 == 4}">
            <geometry><box size="0.1 ${height4} 0.1"/></geometry>
            <origin xyz="0 ${height4/2+0.025} 0" rpy="0 0 0"/>
          </xacro:if>
          <xacro:if value="${p00 == 5}">
            <geometry><box size="0.1 ${height5} 0.1"/></geometry>
            <origin xyz="0 ${height5/2+0.025} 0" rpy="0 0 0"/>
          </xacro:if>
        </visual>
        <xacro:default_inertia mass="0.5"/>
    </link>
<joint name="p00_joint_${name}" type="fixed">
        <parent link="${name}_base"/>
        <child link="p00_block_${name}"/>
        <origin xyz="-0.45 -0.45 0" rpy="1.57 0 0"/>
</joint>

<link name="p01_block_${name}">
    <collision>
        <xacro:if value="${p01 != 1 and p01 != 2 and p01 != 3 and p01 != 4 and p01 != 5}">
            <geometry><box size="0.1 ${height0} 0.1"/></geometry>
            <origin xyz="0 ${height0/2+0.025} 0" rpy="0 0 0"/>
        </xacro:if>
        <xacro:if value="${p01 == 1}">
            <geometry><box size="0.1 ${height1} 0.1"/></geometry>
            <origin xyz="0 ${height1/2+0.025} 0" rpy="0 0 0"/>
        </xacro:if>
        <xacro:if value="${p01 == 2}">
            <geometry><box size="0.1 ${height2} 0.1"/></geometry>
            <origin xyz="0 ${height2/2+0.025} 0" rpy="0 0 0"/>
        </xacro:if>
        <xacro:if value="${p01 == 3}">
            <geometry><box size="0.1 ${height3} 0.1"/></geometry>
            <origin xyz="0 ${height3/2+0.025} 0" rpy="0 0 0"/>
        </xacro:if>
        <xacro:if value="${p01 == 4}">
            <geometry><box size="0.1 ${height4} 0.1"/></geometry>
            <origin xyz="0 ${height4/2+0.025} 0" rpy="0 0 0"/>
        </xacro:if>
        <xacro:if value="${p01 == 5}">
            <geometry><box size="0.1 ${height5} 0.1"/></geometry>
            <origin xyz="0 ${height5/2+0.025} 0" rpy="0 0 0"/>
        </xacro:if>
    </collision>
    <visual>
        <xacro:if value="${p01 != 1 and p01 != 2 and p01 != 3 and p01 != 4 and p01 != 5}">
            <geometry><box size="0.1 ${height0} 0.1"/></geometry>
            <origin xyz="0 ${height0/2+0.025} 0" rpy="0 0 0"/>
```

```
            </xacro:if>
            <xacro:if value="${p01 == 1}">
               <geometry><box size="0.1 ${height1} 0.1"/></geometry>
               <origin xyz="0 ${height1/2+0.025} 0" rpy="0 0 0"/>
            </xacro:if>
            <xacro:if value="${p01 == 2}">
               <geometry><box size="0.1 ${height2} 0.1"/></geometry>
               <origin xyz="0 ${height2/2+0.025} 0" rpy="0 0 0"/>
            </xacro:if>
            <xacro:if value="${p01 == 3}">
               <geometry><box size="0.1 ${height3} 0.1"/></geometry>
               <origin xyz="0 ${height3/2+0.025} 0" rpy="0 0 0"/>
            </xacro:if>
            <xacro:if value="${p01 == 4}">
               <geometry><box size="0.1 ${height4} 0.1"/></geometry>
               <origin xyz="0 ${height4/2+0.025} 0" rpy="0 0 0"/>
            </xacro:if>
            <xacro:if value="${p01 == 5}">
               <geometry><box size="0.1 ${height5} 0.1"/></geometry>
               <origin xyz="0 ${height5/2+0.025} 0" rpy="0 0 0"/>
            </xacro:if>
         </visual>
         <xacro:default_inertia mass="0.5"/>
      </link>
      <joint name="p01_joint_${name}" type="fixed">
         <parent link="${name}_base"/>
         <child link="p01_block_${name}"/>
         <origin xyz="-0.45 -0.35 0" rpy="1.57 0 0"/>
      </joint>

 ….
      <xacro:brown link_name="p00_block_${name}"/>
      <xacro:brown link_name="p01_block_${name}"/>
….
</xacro:macro>

<xacro:RSE name="RSE"
p00="0" p01="0" p02="0" p03="0" p04="0" p05="0" p06="0" p07="0" p08="0" p09="0"
p10="0" p11="0" p12="0" p13="0" p14="0" p15="0" p16="0" p17="0" p18="0" p19="0"
p20="0" p21="0" p22="0" p23="0" p24="0" p25="0" p26="0" p27="0" p28="0" p29="0"
p30="0" p31="0" p32="0" p33="0" p34="0" p35="0" p36="0" p37="0" p38="0" p39="0"
p40="1" p41="1" p42="1" p43="1" p44="1" p45="1" p46="1" p47="1" p48="1" p49="1"
p50="1" p51="1" p52="1" p53="1" p54="1" p55="1" p56="1" p57="1" p58="1" p59="1"
p60="1" p61="1" p62="1" p63="1" p64="1" p65="1" p66="1" p67="1" p68="1" p69="1"
p70="0" p71="0" p72="0" p73="0" p74="0" p75="0" p76="0" p77="0" p78="0" p79="0"
p80="0" p81="0" p82="0" p83="0" p84="0" p85="0" p86="0" p87="0" p88="0" p89="0"
p90="0" p91="0" p92="0" p93="0" p94="0" p95="0" p96="0" p97="0" p98="0" p99="0"/>
```

Приложение В

Параметры стека навигации на роботе “Сервосила Инженер”

В1. Параметры локального планировщика - local_planner_params.yaml

```
DWAPlannerROS:
  acc_lim_theta: 10.0
  acc_lim_trans: 0.1
  acc_lim_x: 2.5
  acc_lim_y: 2.5
  angular_sim_granularity: 0.1
  forward_point_distance: 0.325
  goal_distance_bias: 0.8
  max_scaling_factor: 0.2
  max_vel_theta: 1.0
  max_vel_trans: 0.55
  max_vel_x: 0.2
  max_vel_y: 0.0
  min_vel_theta: 0.05
  min_vel_trans: 0.1
  min_vel_x: -0.2
  min_vel_y: 0.0
  occdist_scale: 0.01
  oscillation_reset_angle: 0.2
  oscillation_reset_dist: 0.05
  path_distance_bias: 0.0
  prune_plan: false
  restore_defaults: false
  scaling_speed: 0.25
  sim_granularity: 0.025
  sim_time: 3.0
  stop_time_buffer: 0.2
  theta_stopped_vel: 0.1
  trans_stopped_vel: 0.1
  twirling_scale: 0.0
  use_dwa: true
  vth_samples: 20
  vx_samples: 3
  vy_samples: 10
  xy_goal_tolerance: 0.3
  yaw_goal_tolerance: 0.3
```

В2.Параметры локальной карты стоимости - local_costmap_params.yaml

```
local_costmap:
  global_frame: map
  robot_base_frame: engineer/base_footprint
  update_frequency: 0.25
  publish_frequency: 0.25
  static_map: false
  rolling_window: true
  width: 6.0
  height: 6.0
  resolution: 0.05

obstacle_range: 3.5
raytrace_range: 3.0
footprint: [[-0.36, -0.20], [-0.36, 0.20], [0.26, 0.20], [0.26, -0.20]]
inflation_radius: 0.6
cost_scaling_factor: 1.2

observation_sources: laser_scan_sensor

laser_scan_sensor: {sensor_frame: engineer/hokuyo_link, data_type: LaserScan, topic: /engineer/laser_scan, marking: true, clearing: true}
```

В3. Параметры глобальной карты стоимости - global_costmap_params.yaml

```
global_costmap:
  global_frame: map
  robot_base_frame: engineer/base_footprint
  update_frequency: 0.5
  static_map: false
  rolling_window: true
  width: 15.0
  height: 15.0
  resolution: 0.05

obstacle_range: 3.5
raytrace_range: 3.0
footprint: [[-0.36, -0.20], [-0.36, 0.20], [0.26, 0.20], [0.26, -0.20]]
inflation_radius: 0.6
cost_scaling_factor: 1.2

observation_sources: laser_scan_sensor

laser_scan_sensor: {sensor_frame: engineer/hokuyo_link, data_type: LaserScan, topic: /engineer/laser_scan, marking: true, clearing: true}
```