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 подготовки: 09.04.04 – Программная инженерия

Магистерская программа: Робототехника

ВЫПУСКНАЯ КВАЛИФИКАЦИОННАЯ РАБОТА

АВТОМАТИЗАЦИЯ ЗАДАЧ ИНВЕНТАРИЗАЦИИ СКЛАДСКОГО ПОМЕЩЕНИЯ С ИСПОЛЬЗОВАНИЕМ ЛОКАЛИЗАЦИИ НАЗЕМНОГО РОБОТА-ДОСТАВЩИКА ПРИ ПОМОЩИ ARUCO-МАРКЕРОВ

Обучающийся 2 курса

группы 11-931 Хазетдинов А.Р.

Научный руководитель

PhD (техничекие науки), Магид Е.А.

профессор кафедры

интеллектуальной

робототехники

Абрамский М.М.

Директор ИТИС КФУ

канд. техн. наук

Казань – 2021

СОДЕРЖАНИЕ

ВВЕДЕНИЕ

За последние несколько лет использование автономных робототехнических систем для автоматизации складских помещений значительно увеличилось. Автономные робототехнические системы позволят полностью изменить подход к инвентаризации товаров и сбора заказов. В обозримом будущем робототехника будет продолжать повышать свою роль в решении типичных задач в управлении складских помещений. Роботизированное управление складским помещением - перспективное направление исследований с высокой практической применимостью. Обычно складские помещения представляют собой строго организованную и контролируемую среду с высоким потенциалом для быстрого развертывания роботов. Одной из задач управления складским помещением, которая имеет высокую эффективность развертывания роботов, является инвентаризация товаров. Инвентаризация складского помещения требует значительного количества ресурсов, поскольку обычно такие складские помещения занимают большие площади, которые заполнены стеллажами высотой 10 и более метров и хранят тысячи товаров. Помимо огромных затрат времени, задача инвентаризации сопряжена с высоким риском травм персонала. Например, из-за необходимости подниматься по складной лестнице, чтобы добраться до большинства полок стеллажей. В таких условиях автономные роботы позволяют проводить инвентаризацию быстрее, чаще и точнее, чем это могли бы сделать люди, одновременно повышая безопасность складского персонала [1]. Основные проблемы, которые решают автономные робототехнические системы:

- Снижение доступности рабочей силы на рынке
- Рост затрат на заработные платы сотрудников
- Человеческий фактор
- И другие.

В данной работе будут использоваться мобильный робот TIAGo Base [2] от компании PAL-Robotics [3] и беспилотный летальный аппарат (БЛА) на базе Pixhawk [4], оснащенный RFID-считывателем [5], которые поддерживают использование Robot Operating System (ROS) [6]. Чтобы гарантировать надежность роботизированной системы, ее необходимо тщательно протестировать перед развертыванием. Первые этапы разработки и тестирования выполнены в виде моделирования, поскольку они обеспечивают ресурсосберегающий способ оценки нового оборудования и программного обеспечения. Несмотря на то, что существует широкий спектр трехмерных симуляторов для робототехнических систем, в последнее время Gazebo [7] стал одним из самых популярных симуляторов благодаря его интеграции с фреймворком Robot Operating System (ROS). Он имеет тесную интеграцию с множеством современных роботов, их трехмерными моделями и встроенными функциями навигации.

Данная работа состоит из нескольких разделов:

1) Обзор литературы. Изучение и анализ научных работ по тематике автоматизации задач складского помещения с использованием автономных робототехнических систем.

2) Складская робототехника. Данный раздел включает определение понятий и перечня решаемых задач.

3) Обзор используемых роботов. Изучение БЛА на базе Pixhawk и мобильного робота TIAGo Base.

4) Моделирование складского помещения. Моделирование и организация складского помещения по стандартам ГОСТ для использования в трехмерном динамическом симуляторе Gazebo.

5) Создание системы автоматизации задач инвентаризации складского помещения с использованием локализации наземного робота-доставщика при помощи ArUco-маркеров. Использование БЛА на базе Pixhawk для автоматического сканирования RFID-меток.

Использование мобильного робота TIAGo Base для навигации в складском помещении и локализации используя ArUco-маркеры.

6) Заключительная часть. Проверка работоспособности всей системы в складском помещении, разработанном для трехмерного динамического симулятора Gazebo.

7) Выводы. Выводы о проделанной работе и будущие улучшения.

1. ОБЗОР ЛИТЕРАТУРЫ

В работе [8] авторы представили подход использования БЛА для инвентаризации товаров в складском помещении. БЛА производит локализацию и картографирование на основе 3D-лидара, который может надежно справиться со сложными ситуациями. Лидар также является основой низкоуровневого механизма обхода препятствий. Кроме того, робот имеет сенсор для идентификации хранимого товара с помощью координатных меток и меток радиочастотной идентификации (англ. Radio Frequency Identification, RFID). Полетное задание обеспечивается системой управления складом (англ. Warehouse Management System, WMS) в виде последовательности складских стеллажей, которые необходимо проверить. Для навигации используется карта складского помещения, которая содержит приблизительное размещение всех стеллажей, а также количество и относительное положение складских товаров. Карта, построенная при помощи 3D-лидара, определяет необходимые позиции, которые БЛА посещает во время выполнения задания.

В статье [9] авторы предлагают новый метод локализации двухмерного штрих-кода на основе ИК-излучения для эффективного планирования пути навигации БЛА. Для эффективного сокращения области поиска они сначала применяют метод выборочного поиска к изображениям на основе ИК-излучения и генерируют множество областей-кандидатов. Расстояние между дроном и целевой точкой в складском помещении, а также формы целевых 2D-штрих-кодов уже известны. Из группы кандидатов успешно отбираются лучшие кандидаты на основе предварительных предположений, полученных заранее. Чтобы извлечь характерную особенность из целевых двумерных штрих-кодов ИК-изображения, они используют различные визуальные функции, начиная от функции локального двоичного шаблона (англ. Local

Binary Pattern, LBP) и заканчивая функцией гистограммы градиента (англ. Histogram of Oriented Gradients или HOG). Метод опорных векторов (англ. Support Vector Machine или SVM) используется для распознавания двухмерных штрих-кодов на основе визуальных характеристик.

В статье [10] автор предлагает использовать мобильного робота с RFID-считывателем, называемого LibBot, для автоматизации задач инвентаризации книг на полках, а также поиск и обнаружение потерянных книг. Хотя до полной автоматизации процесса реорганизации библиотеки еще далеко. Они показывают, что мобильный робот может быть использован для обнаружения книг с помощью, встроенных в них RFID-меток. Таким образом, больше нет необходимости сканировать полки вручную, но товары можно локализовать автоматически через частые промежутки времени, например, каждую ночь. Система LibBot состоит из двух функций, объединенных в один полезный инструмент; способность перемещать и отслеживать движения, выполняемые платформой мобильного робота, а также способность распознавать книги библиотеки с помощью RFID-считывателя. Объединение информации, полученной с помощью обеих этих функций, позволяет локализовать RFID-метки и книги, которые они идентифицируют.

В [11] авторами представлена реализация мобильного робота, который может автономно выполнять инвентаризацию. Робот может использовать предоставленную карту или построить свою собственную с нуля. Он оснащен сканером RFID, а также датчиками и исполнительными устройствами, которые позволяют им покрывать заданное пространство с максимальной эффективностью и без вмешательства человека. Ключевой особенностью алгоритма навигации является способность разумно планировать пути, по которым робот приближается к областям, подлежащим сканированию, вместо того чтобы тратить время на открытое пространство. В некотором смысле это дополнительное поведение к хорошо изученной классической проблеме

покрытия в мобильной робототехнике. Последний будет использоваться, например, в роботизированном пылесосе или автономном сельскохозяйственном тракторе. В данном случае они хотят направлять робота к стеллажам, которые содержат товары, при этом минимизируя время и энергию, затрачиваемые на прохождение открытых площадей. Эта технология дает возможность проводить инвентаризацию гораздо чаще, чем это было бы во время ручных операций.

В статье [12] автор предлагает новый подход к инвентаризации на открытом воздухе с использованием БЛА и RFID, чтобы преодолеть некоторые ограничения устаревших подходов. Разрабатываемая система состоит из трех основных компонентов. Первый — это RFID-считыватель, который обнаруживает RFID-метки продуктов, распределенных на открытом складе. Он хранит идентификатор обнаруженного тега, местоположение, время и количество продуктов. Второй — это БЛА, который перемещается по всему двору вручную или автоматически. Последняя — это серверная программа для инвентаризации, которая сравнивает значения из базы данных и реальные данные о товарах.

Работа [13] представляет автоматизированную система для подтверждения количества товарно-материальных запасов, зарегистрированных во время поступления или отгрузки товара, путем развертывания мобильных RFID-считывателей. Эти считыватели переносятся БЛА для проверки инвентаря после того, как товар был перемещен на стеллажи. Данный метод снижает вероятность неучтенных запасов. Хотя существуют роботы, предназначенные для покрытия двумерного пространства, такие как Roomba iRobot, ни один из них не был разработан для покрытия трехмерного пространства, необходимого для отслеживания товаров. Неровная местность на складе и предметы, хранящиеся на более высоких местах, недоступные для наземного робота, вызывают необходимость в летающей системе. БЛА, оснащенный

RFID-считывателем, может потребоваться для сохранения информации о своей траектории, а не просто запрограммирован для достижения пункта назначения. Кроме того, при проектировании системы БЛА необходимо учитывать динамические препятствия. Система с несколькими БЛА обеспечивает гибкость, позволяющую приспосабливаться к изменяющейся планировке склада, чего нет при развертывании статического RFID-считывателя. Помимо выполнения инвентаризации, система с несколькими БЛА также позволит отслеживать местоположение товара на складе.

Для обеспечения безопасной и успешной проверки запасов в складских помещениях в работе [14] автор предлагает мультисенсорную платформу слияния на основе расширенного фильтра Калмана (EKF) с использованием недорогих датчиков: трех камер, двухмерного лидара, одномерного датчика дальности и IMU. Автор использует простую карту, которая состоит из информации о позиции прикрепленных координатных меток, поэтому нет необходимости выполнять пилотируемые полеты для построения карты местности. Кроме того, автор предлагает три метода слияния данных:

1) Отклонение выбросов с использованием стандартного компонентного теста Махаланобиса

2) Добавление визуального SLAM к обновлению результатов измерений путем введения «псевдоковариации»

3) Распознавание полос на полу - для преодоления проблем смещения, что приводит к повышению надежности мультисенсорной платформы.

В работе [15] предлагается автономный подход к инвентаризации складского помещения, в котором используются БЛА, которые, как предполагается, выполняют непрерывную проверку товаров в складском помещении. Предлагаемое решение должно быть эффективным с вычислительной точки зрения, а также обеспечивать уровень точности, соответствующий эффективному управлению складом. БЛА использует

легкие модели сверточной нейронной сети (CNN) для обнаружения и распознавания в реальном времени этикеток на упаковках во время воздушного сканирования окружающей среды. Летательный аппарат с бортовым датчиком окружающей среды, состоящим из четырех обычных RGB-камер и достаточной вычислительной мощностью, может автономно проверять складские помещения и выполнять задачи компьютерного зрения для отслеживания этикеток / штрих-кодов на упаковках. Результаты экспериментов, полученные на видеосъемке реальной складской среды, продемонстрировали осуществимость предложенного решения и его надежное использование для выполнения логистических задач. Стоит отметить, что предлагаемый подход не направлен на распознавание текста или кодов на этикетках упаковки, поскольку для этой задачи доступны хорошо отработанные и надежные процедуры.

В статье [16] авторы демонстрируют современную систему инвентаризации на базе БЛА, используя как теоретические, так и практические материалы. Во-первых, авторы предлагают общую архитектуру для роя БЛА, которые отвечают за идентификацию товаров на стеллажах с помощью технологии QR-кода. Программно-аппаратная архитектура включает в себя компоненты управления роем БЛА, планирование пути БЛА, беспроводную зарядку, хранение и проверку данных инвентаризации. Во-вторых, авторы описывают экспериментальную реализацию предложенной архитектуры с помощью недорогих мини-БЛА в целевом сценарии использования, связанном с продовольственным складским помещением. Также они анализируют компромисс между точностью распознавания QR-кода и временем завершения инвентаризации (то есть время, необходимое для удержания высоты на каждой полке), и предложили алгоритм для настройки параметров мобильности БЛА, таких как скорость и количество попыток сканирования для каждого стеллажа.

Автор представляет в статье [17] новую схему складской инвентаризации. Основная цель этой работы - сделать процесс инвентаризации полностью автономным. С этой целью беспилотный наземный робот (БНР) и БЛА работают совместно. БНР используется в качестве несущей платформы и считается наземным ориентиром для полета БЛА в помещении. БЛА используется как мобильный сканер. БНР перемещается среди рядов стеллажей с БЛА. У каждого сканируемого стеллажа БНР останавливается, и БЛА взлетает, чтобы летать вертикально, сканируя товары на этом стеллаже. Как только БЛА окажется наверху, БНР перемещается к следующему стеллажу, и, поскольку БЛА принимает БНР в качестве наземного ориентира, он будет следовать за ним автономно. Процесс повторяется до полного сканирования ряда стеллажей. Затем БЛА приземляется на БНР и заряжает свой аккумулятор, в то время как БНР перемещается к следующему ряду стеллажей.

В статье [18] автор использует БЛА на базе ROS для автоматизации задач инвентаризации в складском помещении. Для этого они построили БЛА с мультимодальными всенаправленными датчиками, быстрым бортовым компьютером и надежным каналом передачи данных. Датчики включают в себя легкий 3D-лидар, три стереокамеры и модуль считывания радиочастотной идентификации (RFID). Все компоненты легкие и поэтому хорошо подходят для БЛА. БЛА может локализоваться в помещениях, сочетая визуальную одометрию и данные лидара с полученной 3D-картой. Он избегает статических и динамических препятствий, обнаруживаемых бортовыми датчиками. В БЛА они используют ROS Indigo в качестве промежуточного программного обеспечения поверх Ubuntu 14.04, чтобы облегчить быструю разработку за счет модульной конструкции программного обеспечения. Это позволяет передавать технологии между несколькими БЛА. Кроме того, ROS позволяет легко и гибко подключаться к нескольким наземным станциям управления.

Автономная робототехническая система [19] состоит из наземного робота и БЛА. Основная задача наземного робота - определение его координат относительно окружающих предметов на складе. Такой подход позволяет точно рассчитать положение каждого поддона. Для определения местоположения поддона в 2D пространстве робот использует алгоритм SLAM, основанный на графах. Для построения карты исследуемой территории робот использует LIDAR. Робот получает двухмерные сканы, а затем сравнивает их с локальными данными. Таким образом, робот находит свою позицию и дополняет карту новыми препятствиями. После этого наземный робот объединяет подкарты в одну карту и находит окончательную карту местности, используя методы нелинейной оптимизации. Для системы локализации БЛА они используют камеру на наземном роботе и два концентрических шаблона активных ИК (инфракрасных) маркеров на БЛА. Цель БЛА - полет в небольшой цилиндрической рабочей зоне над мобильным роботом. Мы измеряем высоту БЛА, используя распознанную ИК-диаграмму.

Для инвентаризации складского помещения был разработан наземный робот [20]. Чтобы построить робота, авторы установили на шасси четыре мотора. Двигатели имеют возможность независимого управления драйверами. Arduino используется для передачи команд драйверам двигателей относительно того, как двигатели должны вращаться. Arduino также управляет двумя серводвигателями, установленными спереди и сзади, и двумя ультразвуковыми датчиками, установленными на сервоприводах. Команда на перемещение робота дается Arduino с помощью Raspberry Pi. Raspberry Pi также получает данные от четырехколесных энкодеров, каждый из которых установлен на двигателе, и инерциального измерительного блока. Raspberry Pi отправляет эти данные на сервер. Сервер также получает данные с внешней веб-камеры, установленной на роботе.

Авторы использовали внешнюю камеру, установленную на потолке. Это должно хорошо работать на складе, поскольку они могут установить камеры на потолке или использовать поток с камер видеонаблюдения. Поток с этой камеры используется сервером для обнаружения робота с помощью методов обработки изображений. Чтобы обнаружить робота и препятствия, они наклеили на них цветные наклейки. После обнаружения роботов они могут получить начальную и конечную точки пути. А с помощью алгоритмов поиска и планирования мы можем указать роботу путь, по которому он сможет двигаться дальше. Они отправляют команду на основе этого пути, и он движется. Когда робот подъезжает достаточно близко к цели, он останавливается. Потолочные камеры также можно использовать для инвентаризации и учета.

2. СКЛАДСКАЯ РОБОТОТЕХНИКА

Складская робототехника — это использование автоматизированных систем, роботов и специализированного программного обеспечения для транспортировки материалов, выполнения различных задач и оптимизации / автоматизации складских процессов. В последние годы робототехника стала популярной в средах управления цепочками поставок, распределительными центрами и складами и продолжает играть значительную роль в автоматизации складов [21].

Технологические достижения и растущая конкуренция вынуждает задуматься об использовании робототехники в складских помещениях. Благодаря своей способности повышать производительность, точность и эффективность работы складские роботы стали незаменимыми для эффективных складских операций. Автоматизация складских помещений повышает ценность складских операций, автоматизируя выполнение второстепенных, повторяющихся задач, позволяя тем самым людям сосредоточиться на более сложных задачах. Самые распространенные типы складской робототехники:

1) Автоматизированные транспортные средства
2) Автоматизированные системы хранения и поиска
3) Коллаборативные наземные роботы
4) Роботизированные манипуляторы

Автоматизированные транспортные средства (рис.2.1) помогают транспортировать расходные материалы и инвентарь в пределах складских помещений. Автоматизированные транспортные средства используются в операциях для замены вилочных погрузчиков с ручным приводом или тележек для перевоза груза. Некоторые автоматизированные транспортные средства автономно перемещаются по складским помещениям, следуя

установленным маршрутам, магнитными полосами, дорожками, датчиками, встроенными в пол. В других автоматизированные транспортные средства используются камеры, лидары, инфракрасные лучи и другие передовые технологии для навигации по путевым точкам и для определения препятствий и предотвращения столкновений.

Рис.2.1. Автоматизированные транспортные средства[1].

Автоматизированные системы хранения и поиска относится к группе систем с компьютерным управлением, которые помогают автоматизировать управление запасами и хранить / получать товары по запросу. Эти системы предназначены для облегчения быстрого поиска и размещения продуктов и обычно сочетаются с программным обеспечением для складских операций. Они работают либо как подъемные краны, либо как челноки на фиксированных путях и могут легко перемещаться по горизонтали по проходам с продуктами и по вертикали для укладки или удаления предметов. Автоматизированные системы хранения и поиска используются в складских помещениях для ускорения выполнения заказов и погрузочно-разгрузочных операций.

Коллаборативные мобильные роботы (рис.2.2) — это полуавтономные мобильные роботы, предназначенные для того, чтобы помогать людям

[1] Фото взято с https://taknek.com/wp-content/uploads/2021/05/agv-e1621265674631.jpg

выполнять разнообразные задачи в складском помещении. Некоторые коллаборативные мобильные роботы следуют за людьми-сборщиками по всему складскому помещению и действуют как мобильные складские места для собранных заказов. Коллаборативные мобильные роботы оснащены датчиками, которые позволяют им различать препятствия и ящики, обеспечивая тем самым точную навигацию по объекту. Они также могут помочь ускорить выполнение заказов, доставив выбранные заказы работникам в других местах на складе, например на сортировочных или упаковочных станциях.

Рис.2.2. Коллаборативные мобильные роботы[2].

Роботизированные манипуляторы (рис.2.3), тип роботов для подбора и размещения, представляют собой многосуставные конструкцию, используемые для манипулирования товарами в распределительных центрах и в складских помещениях. Поскольку роботизированные манипуляторы

[2] фото взято с https://www.technische-logistik.net/sites/default/files/styles/aktuelles/public/bilder/6Rivers_Unternehmen_TL0420.jpg?itok=l8hPq4J4

могут перемещать, поворачивать и поднимать предметы, их можно использовать в следующих складских операциях:

- Сборка
- Упаковка
- Выдача
- Хранение
- Паллетирование

Рис. 2.3. Роботизированные манипуляторы[3].

Современные компании используют складскую робототехнику, чтобы улучшить выполнение заказов и лучше управлять запасами, оставаясь при этом впереди конкурентов. Складская робототехника помогает сократить

[3] Фото взято с https://6river.com/wp-content/uploads/2019/11/pick-and-place-robot-min.webp

время и ресурсы, затрачиваемые на извлечение и транспортировку товаров по складскому помещению, позволяя работникам сосредоточиться на более сложных процессах, таких как упаковка и отгрузка заказов. Хотя повышение эффективности и снижение затрат являются очевидными преимуществами развертывания роботов в складском помещении, есть несколько дополнительных преимуществ, которые также следует учитывать:

- Высокая точность
- Снижение физической и умственной нагрузки
- Автоматизация рутинных задач

Инвестиции в складскую робототехнику приводят к повышению производительности и точности управления запасами. Человеческие ошибки могут быть довольно дорогостоящими для бизнеса и могут привести к снижению удовлетворенности клиентов. Складская робототехника с высокой точностью автоматизирует обработку, сбор, сортировку и пополнение товаров, что гарантирует высокий уровень точности при выполнении заказов и складских операциях.

Роботы берут на себя трудоемкие, стрессовые и опасные аспекты складских операций, такие как транспортировка товара. Беря на себя некоторые опасные и стрессовые задачи, складские роботы снижают физическую и умственную нагрузку на рабочих.

Складские помещения могут повысить эффективность, снизить количество ошибок и более точно выполнять заказы за счет развертывания подходящих роботов. Складские роботы автоматизируют рутинные задачи, чтобы люди могли сосредоточить свое время и энергию на выполнении сложных задач, требующих мелкой моторики или критического мышления.

Роботы помогают уменьшить/устранить ошибки, ускорить выполнение заказов, сократить расходы и улучшить управление запасами.

2.1. АВТОМАТИЗАЦИЯ ЗАДАЧ ИНВЕНТАРИЗАЦИИ СКЛАДСКОГО ПОМЕЩЕНИЯ С ИСПОЛЬЗОВАНИЕМ ЛОКАЛИЗАЦИИ НАЗЕМНОГО РОБОТА-ДОСТАВЩИКА ПРИ ПОМОЩИ ARUCO-МАРКЕРОВ

Одной из важных задач, которые связаны с работой со складскими помещениями, является инвентаризация. Когда складское помещение относится к крупным складским предприятиям, необходимо, чтобы инвентаризация товара проходила с высокой скоростью. Инвентаризация товаров в складских помещениях, где стеллажи могут достигать нескольких метров высоты и содержать несколько тысяч товаров, может вызвать задержки в обновлении информации из-за затрат времени на выполнение поставленной задачи, а также из-за возможных травм рабочих, выполняющих инвентаризацию объектов, находящихся на больших высотах. В данной дипломной работе будет описано решение автономной инвентаризации складского помещения с использованием локализации наземного робота-доставщика при помощи ArUco-маркеров.

Благодаря мобильности и возможности использования дополнительного оборудования автономная робототехническая система является идеальным кандидатом для выполнения задач, связанных с инвентаризацией складского помещения. Они помогут производить инвентаризацию быстрее, качественнее и эффективнее, чем люди.

В данной работе были использованы два типа роботов: наземные и летающие. В качестве наземного робота-доставщика был использован мобильный робот TIAGo Base от компании PAL-Robotics. TIAGo Base оборудован всеми необходимыми сенсорами для выполнения роли робота-доставщика. Дополнительно TIAGo Base оборудован камерой для дополнительной локализации при помощи ArUco-маркеров [22]. БЛА на базе

Pixhawk, оборудованный RFID-считывателем, сканирует стеллажи складского помещения для инвентаризации товаров. Основная задача разрабатываемой системы заключается в следующем. Каждый стеллаж складского помещения оборудован уникальным ArUco-маркером. Мобильный робот TIAGo Base получает список путевых точек, где ему необходимо остановиться. При обнаружении ArUco-маркера мобильный робот TIAGo Base начинает процесс выравнивания относительно него и обновляет свою текущую позицию используя данные, полученные о расположении данного ArUco-маркера на карте складского помещения. После успешного выравнивания мобильного робота TIAGo Base относительно обнаруженного ArUco-маркера, БЛА на базе Pixhawk, оборудованный RFID-считывателем, взлетает и считывает обнаруженные RFID-метки, установленные на складских товарах. После сканирования RFID-меток БЛА на базе Pixhawk начинает посадку на мобильного робота TIAGo Base. Затем мобильный робот TIAGo Base получает следующую путевую точку из своего списка путевых точек и начинает движение к ней, пока не будет достигнута последняя путевая точка из списка путевых точек.

3. ОБЗОР ИСПОЛЬЗУЕМЫХ РОБОТОВ И ИНСТРУМЕНТОВ

3.1. ROBOT OPERATING SYSTEM (ROS)

Robot Operating System (ROS) - это фреймворк для создания программного обеспечения для роботов. Его полезность не ограничивается роботами, но большинство предоставляемых инструментов ориентировано на работу с периферийным оборудованием.

ROS разделена на более чем 2000 пакетов, каждый из которых обеспечивает специализированную функциональность. ROS предоставляет

функциональные возможности для абстракции оборудования, драйверов устройств, обмена данными между процессами на нескольких машинах, инструментов для тестирования и визуализации и многого другого.

Ключевой особенностью ROS является способ запуска программного обеспечения и способ взаимодействия, позволяющий разрабатывать сложное программное обеспечение. ROS позволяет соединить сеть процессов (узлов) с центральным контроллером. Узлы можно запускать на нескольких устройствах, и они могут подключаться к этому контроллеру различными способами.

Основные способы создания сети — это предоставление запрашиваемых услуг или определение соединений издателя / подписчика с другими узлами. Оба метода обмениваются данными через указанные типы сообщений. Некоторые типы предоставляются базовыми пакетами, но также возможно создание своих типов данных. ROS ориентирована на Unix-подобную систему: Ubuntu, Debian полностью поддерживаемые, Windows, Gentoo и MAC OS X экспериментальные версии.

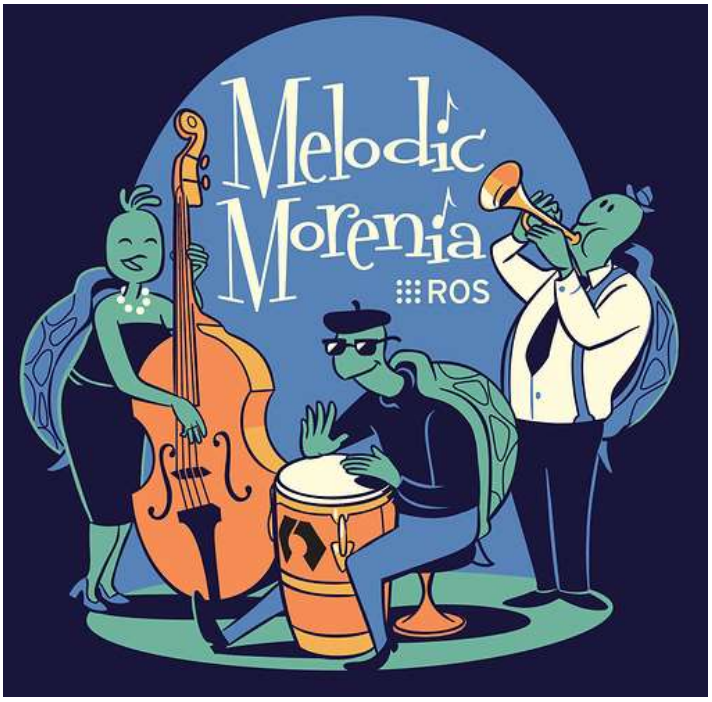


Рис. 3.1.1. Одна из версий ROS - Melodic Morenia[4].

3.2. СИМУЛЯТОР GAZEBO

Gazebo (рис.3.2.1) - это бесплатная среда для моделирования роботов с открытым исходным кодом, разработанная компанией Willow Garage [23]. В качестве многофункционального инструмента для разработчиков робототехнических систем (РТС), Gazebo поддерживает следующее:

- Проектирование моделей роботов
- Быстрое прототипирование и тестирование алгоритмов
- Регрессионное тестирование с использованием реалистичных сценариев
- Моделирование внутренней и наружной среды
- Моделирование данных датчиков для лидаров, 2D / 3D камер, датчиков типа Kinect, контактных датчиков, и др.
- 3D-объекты и сценарии, использующие объектно-ориентированную графику
- Объектно-ориентированный графический движок (OGRE [24])
- Несколько высокопроизводительных физических движков (Open Dynamics Engine (ODE)[25], Bullet [26], Simbody [27] и Dynamic Animation and Robotics Toolkit (DART)[28] для моделирования реальной динамичности.

[4] Фото взято с https://raw.githubusercontent.com/ros-infrastructure/artwork/master/distributions/melodic_with_bg.png

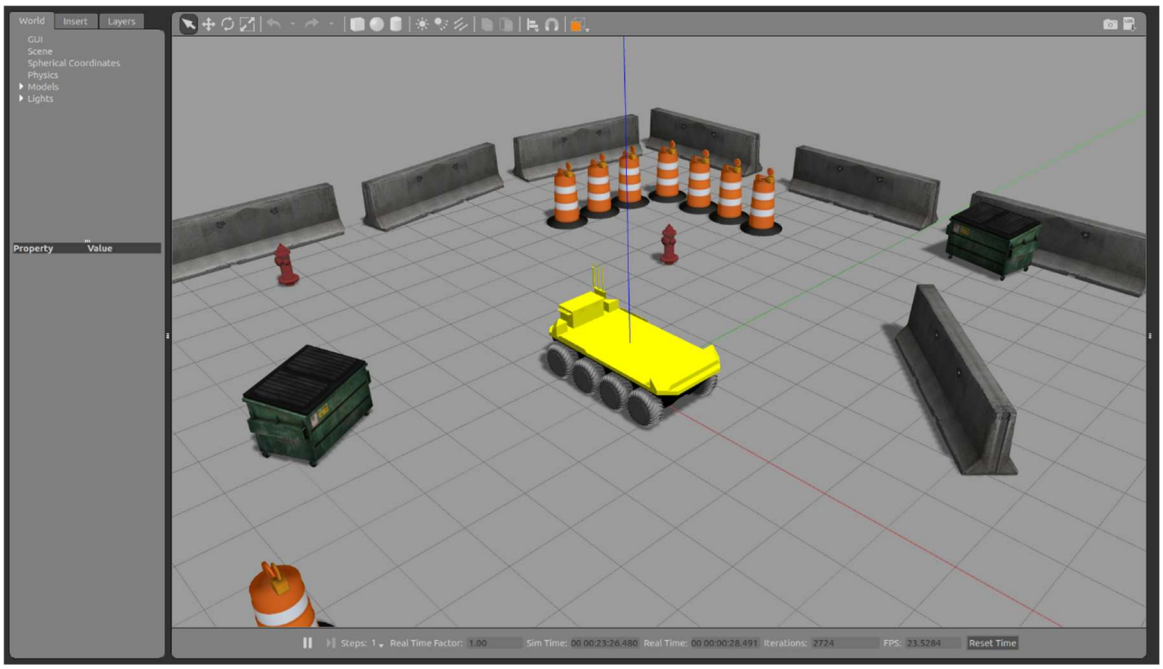


Рис. 3.2.1. Графический интерфейс симулятора Gazebo.

3.3. МОБИЛЬНЫЙ РОБОТ TIAGO BASE

TIAGo Base (3.3.1) — это мобильный робот для помещений, разработанный с учетом минимальной занимаемой площади и максимальной полезной нагрузки. Его основными датчиками являются лидар, IMU и ультразвуковые датчики, хотя можно легко добавить дополнительные устройства и датчики. TIAGo Base имеет встроенные динамики, поэтому робот может сообщить своему окружению о завершении задачи. Этот робот работает внутри ROS Melodic и использует самые современные компоненты ros_control [27].

Рис. 3.3.1. Мобильный робот TIAGo Base (https://robots.ros.org/assets/img/robots/tiago-base/tiago-base_600x600.png).

Мобильный робот TIAGo Base содержит следующие пакеты для работы с ROS [29] и Gazebo:

- Базовая конфигурация
  - Модель робота – pmb2_description
  - Конфигурация контроллера робота - pmb2_controller_configuration
  - Файлы с информацией о роботе - pmb2_bringup
- Аппаратные драйверы и моделирование
  - Аппаратное моделирование робота - pmb2_hardware_gazebo
  - Конфигурация контроллера для моделирования - pmb2_controller_configuration_gazebo
  - Пакеты для работы с Gazebo – pmb2_gazebo

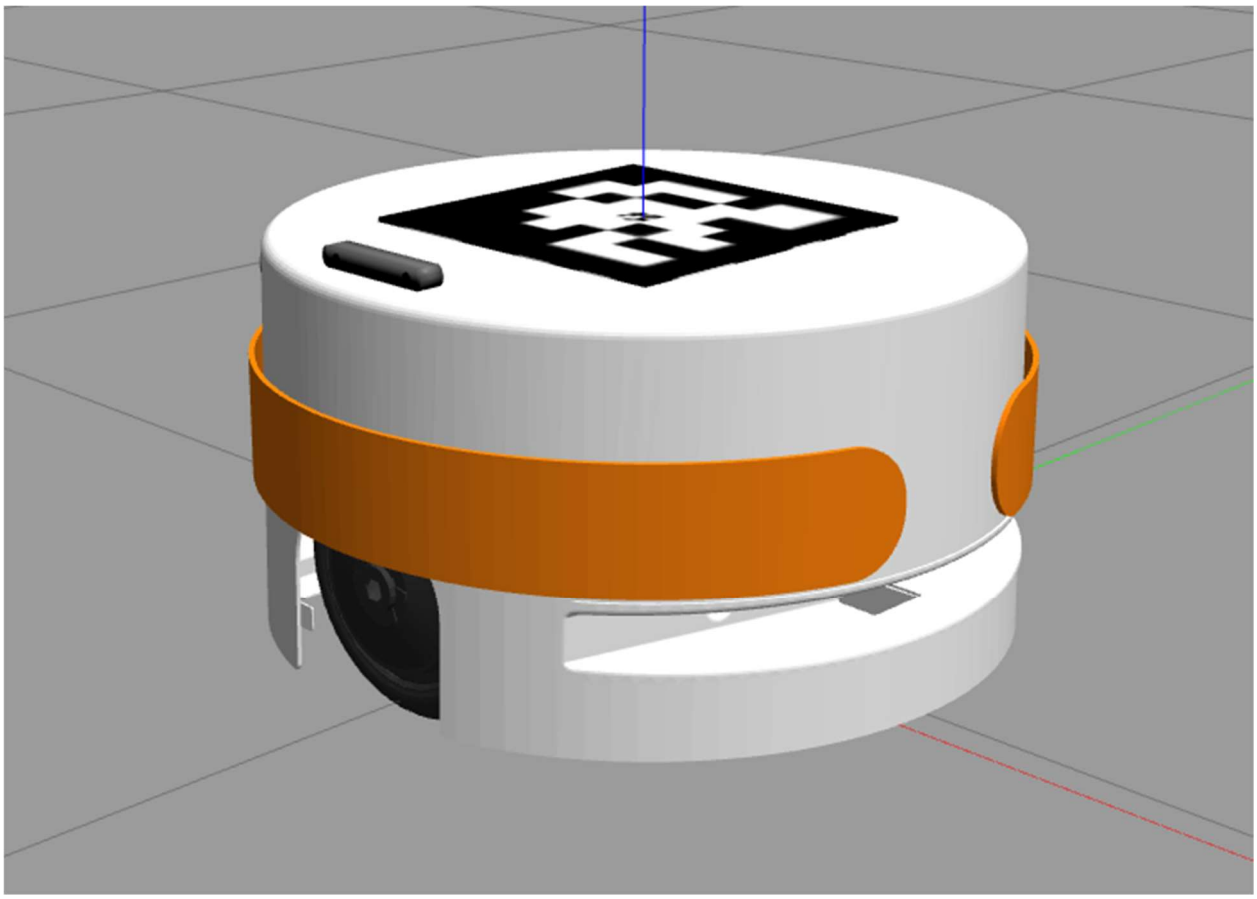


Рис. 3.3.2. Мобильный робот TIAGo Base в симуляторе Gazebo.

3.4. БЕСПИЛОТНЫЙ ЛЕТАТЕЛЬНЫЙ АППАРАТ НА БАЗЕ PIXHAWK

Pixhawk (рис.3.4.1) (также PX4FMU) — открытые аппаратные полетные контроллеры, работающие под управлением PX4 [30] или ArduPilot [31] в ОС NuttX. Имея множество форм-факторов, существуют версии, предназначенные для различных вариантов использования и сегментов рынка.

Рис. 3.4.1. Полетный контроллер Pixhawk (https://www.robostore.com.ua/content/images/25/1000x1000l80nn0/poletnyy-kontroller-ardupilot-pixhawk-2.48-original-57116941999995.jpg).

PX4 — это программное обеспечение для полетного контроллера с открытым исходным кодом. Некоторые из ключевых функций PX4:

- Управляет многими различными рамами / типами транспортных средств, включая: летательные аппараты (мультикоптеры, самолеты с неподвижным крылом и самолеты вертикального взлета и посадки), наземные и подводные транспортные средства.
- Большой выбор различного оборудования, датчиков и другой периферии для использования.
- Различные режимы полета и функции безопасности.

PX4 — это основная часть платформы дронов, которая включает в себя наземную станцию QGroundControl [32], полетный контроллер Pixhawk, и MAVSDK для интеграции с сопутствующими компьютерами, камерами и другим оборудованием с использованием протокола MAVLink [33]. PX4 поддерживается проектом Dronecode Project [34].

Лаборатория интеллектуальных робототехнических систем (ЛИРС) имеет БЛА PX4-LIRS на базе Pixhawk (рис.3.4.2).

Рис. 3.4.2. БЛА PX4-LIRS на базе Pixhawk.

MAVROS (MAVLink + ROS) [35] — это пакет для ROS, предоставляющий возможность управлять БЛА по протоколу MAVLink. MAVROS поддерживает полетные стеки PX4 и APM. Связь организовывается по UART, USB, TCP или UDP. MAVROS подписывается на определенные ROS-топики в ожидании команд, публикует в топики телеметрию, и предоставляет сервисы.

Студенты ЛИРС создали модель БЛА PX4-LIRS для трехмерного симулятора Gazebo (рис.3.4.3).

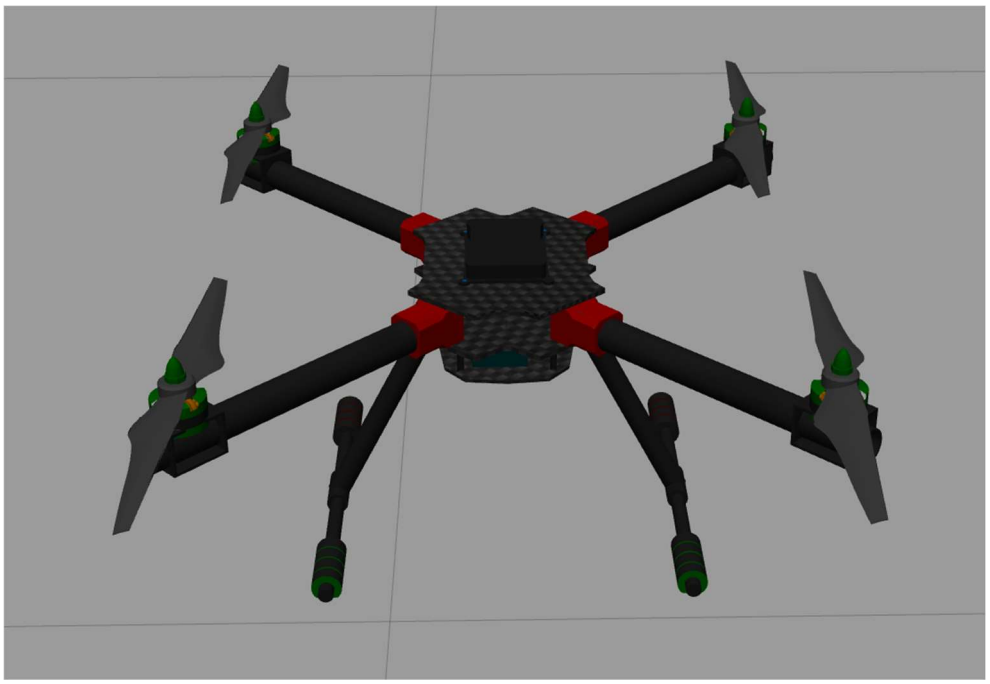


Рис. 3.4.3. БЛА ЛИРС на базе Pixhawk в симуляторе Gazebo.

4. МОДЕЛИРОВАНИЕ СКЛАДСКОГО ПОМЕЩЕНИЯ

Алгоритмы РТС требуют тщательного тестирования и проверки перед фактическим развертыванием, и моделирование - способ обеспечить это. Таким образом, должна быть смоделирована соответствующая среда, чтобы гарантировать качество разработанных роботов и алгоритмов.

4.1. ТИПЫ СКЛАДСКИХ ПОМЕЩЕНИЙ

Компания Knight Frank [36] разработала классификацию складских помещений. Согласно данной классификации, все складские помещения делятся на классы и подклассы:

- Класс A - в свою очередь, делится на подклассы A и A+
- Класс B - также делится на подклассы B и B+
- Классы C и D

Классификация в основном отражает технические характеристики складских зданий и их оборудования.

Класс А и А+ - современное одноэтажное складское здание из легких металлических конструкций и сэндвич-панелей, желательно прямоугольной формы, без колонн или с шагом колонн не менее 12 м для класса А+ и расстоянием между колоннами не менее 9 м для класса А, а также пролетом не менее 24 м. Требуются высокие потолки: не менее 13 м для класса А+ и 10 м для класса А, что позволяет устанавливать многоуровневое стеллажное оборудование.

Рис. 4.1.1. Складское помещения класса А[5].

Класс В и В+ - одноэтажное складское здание, желательно прямоугольной формы, недавно построенное или реконструированное. Также требуются высокие потолки: не менее 8 м для класса В+ и 6 м для класса В.

[5] Фото взято с https://lh3.googleusercontent.com/proxy/jgrpwZIMNDjvg_mtNNl5af7GFejbScyaSyzDF8ccgtk57ym9EioNF96qAby6zyiW4HTJyA_a0rQ0-qLcxedE

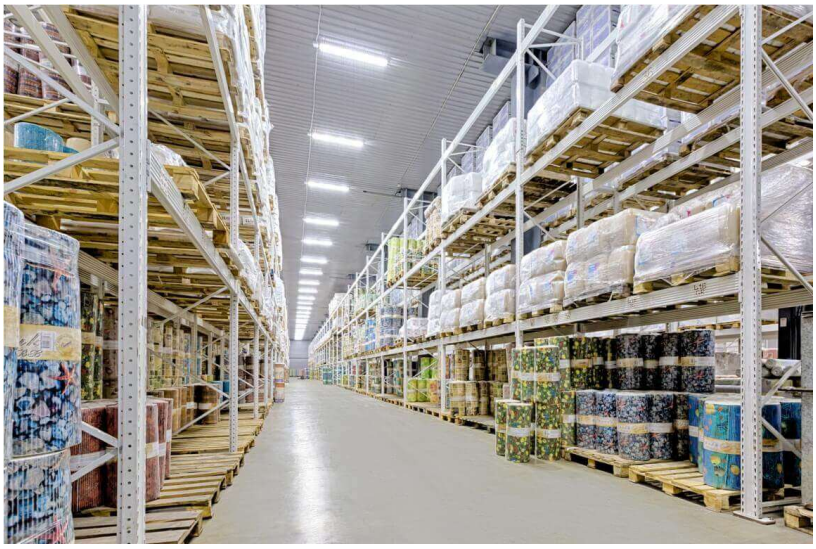


Рис. 4.1.2. Складское помещения класса B[6].

Класс C - капитальное производственное помещение или утепленный ангар с минимальной высотой потолков 4 м. Класс D — это подвал, неотапливаемые производственные помещения или ангары.

Рис. 4.1.3. Складское помещения класса C[7].

[6] Фото взято с https://skperspektiva.ru/assets/files/uploads/images/Sklad1.jpg
[7] Фото взято с http://newsklad.ru/images/pages/img/sklad_klass_c2.jpg

4.2. ТЕХНИЧЕСКИЕ ХАРАКТЕРИСТИКИ СТЕЛЛАЖА

Согласно ГОСТ Р 55525-2013 [37] стеллажи делятся на следующие типы:

- Фронтальный (рис.4.2.1)
- Набивной (рис.4.2.2)
- Консольный (рис.4.2.3)

У разных типов стеллажей разное предназначение. Например, набивные (или глубокие) стеллажи позволяют штабелировать несколько товаров на одном уровне стеллажа, проталкивая товары глубже в стеллаж каждый раз, когда необходимо загрузить груз. Таким образом, для немедленной выгрузки и передачи доступен только последний загруженный груз. Таким образом, такие стеллажи удобны, когда товары идентичны. Этот тип стеллажа позволяет эффективно использовать складские площади.

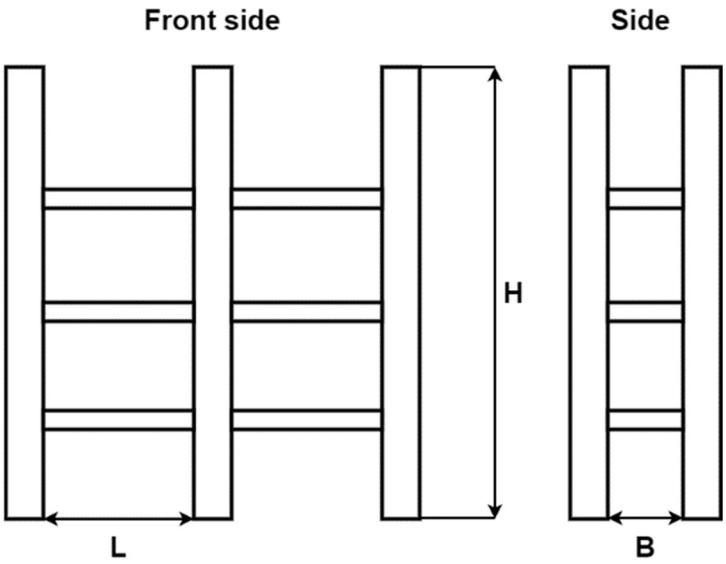


Рис. 4.2.1. Фронтальный стеллаж.

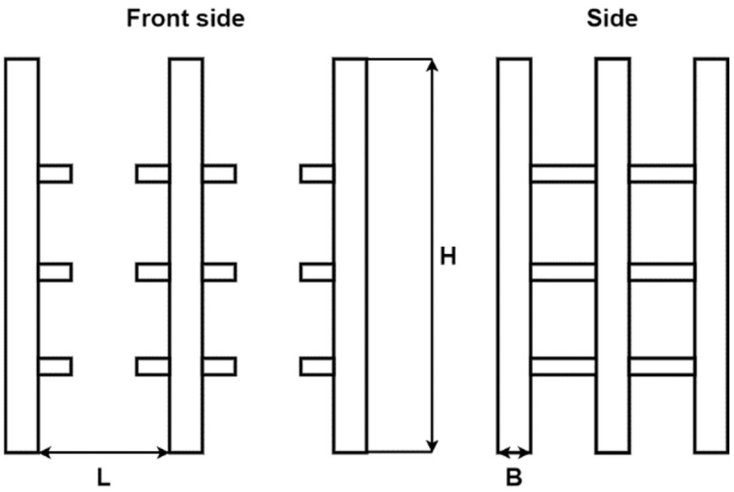


Рис. 4.2.2. Набивной стеллаж.

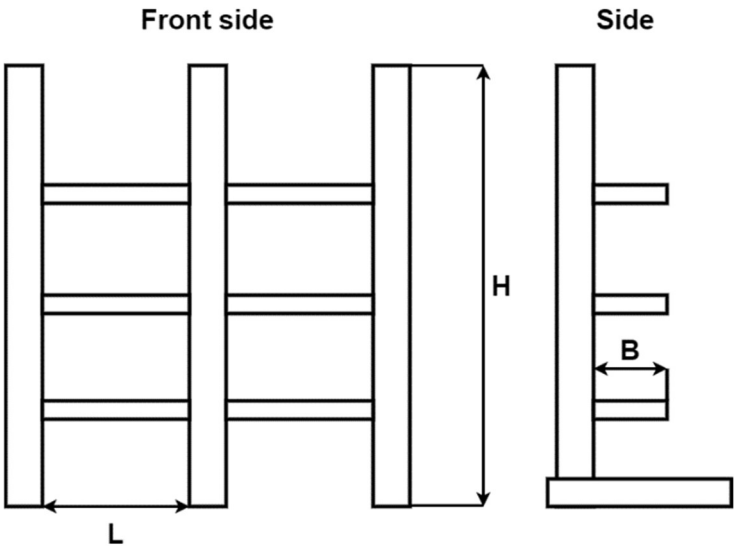


Рис.4.2.3. Консольный стеллаж.

Консольные стеллажи имеют более простую конструкцию, однако их грузоподъемность ограничена. Фронтальные стеллажи обычно используются на коммерческих складах благодаря своей прочности и простоте инвентаризации товаров. Товары, загруженные на передние стеллажи, видны из проходов, а инвентаризация может быть произведена с использованием QR-кодов, RFID-меток и т.д.

4.2.1. СХЕМА СТЕЛЛАЖА

Нормы, введенные ГОСТ 16140-77 [38], определяют минимальное расстояние между стеллажами и стеной складского помещения (включая проход, необходимый для движения погрузчика) - оно должно быть не менее 0,7 м. Этой ширины достаточно, чтобы вдоль стеллажа могли проходить люди и погрузочное оборудование. Монтаж систем фронтальной загрузки регламентируется требованиями ГОСТ Р 55525-2013 и осуществляется следующими способами:

• Широкий проход - ширина варьируется от 2,5 до 3,7 м; площадь коридора позволяет погрузчику свободно разворачиваться.

• Узкий проход - размеры от 1,5 до 1,9 м, транспортный проход должен использоваться для разворота транспорта.

4.2.2. РАСПОЛОЖЕНИЕ ПОДДОНОВ

Размещение поддонов на стеллажах также регулируется требованиями ГОСТ Р 55525-2013. В таблице 4.2.2.1 приведены минимальные значения технологических зазоров в зависимости от высоты груза. Схема на рис. 4.2.2.1 определяет размещение поддонов на стеллажах.

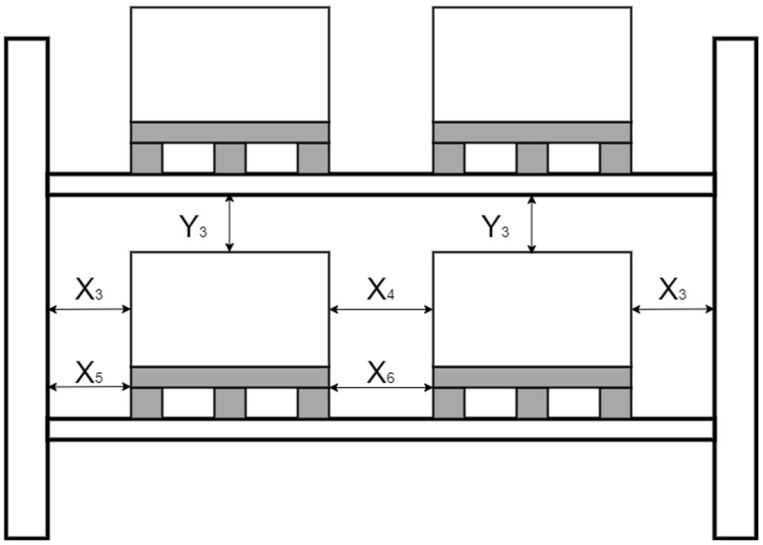


Рис. 4.2.2.1. Размещение поддонов на стеллаже.

Таблица 4.2.2.1. Минимальные значения технологических зазоров в зависимости от высоты груза

Высота размещения груза H, мм	Система хранения в широких проходах		Система хранения в узких проходах			
			Класс А		Класс В	
	X3, X4, X5, X6, мм	Y3, mm	X3, X4, X5, X6, мм	Y3, мм	X3, X4, X5, X6, мм	Y3, мм
3000	75	75	75	75	75	75
6000	75	100	75	75	75	100
9000	75	125	75	75	75	125
12000	75	150	75	75	100	150
15000	75	175	75	75	100	175

4.3. МОДЕЛИРОВАНИЕ СКЛАДСКОГО ПОМЕЩЕНИЯ ДЛЯ СИМУЛЯТОРА GAZEBO

4.3.1. СКЛАДСКОЕ ПОМЕЩЕНИЕ AWS ROBOTICS

AWS Robotics [39] недавно представила модель складского помещения для симулятора Gazebo. Склад имеет размер 14x21 м и имеет 4 больших стеллажа и 6 малых стеллажей (рис. 4.3.1.1). К сожалению, эта среда является небольшой и не соответствует правилам хранения грузов. Его можно использовать для моделирования в ограниченном количестве ситуаций.

Этот случай доказывает, что надлежащее моделирование склада по-прежнему необходимо для коммерческих целей и имеет значительную практическую ценность.

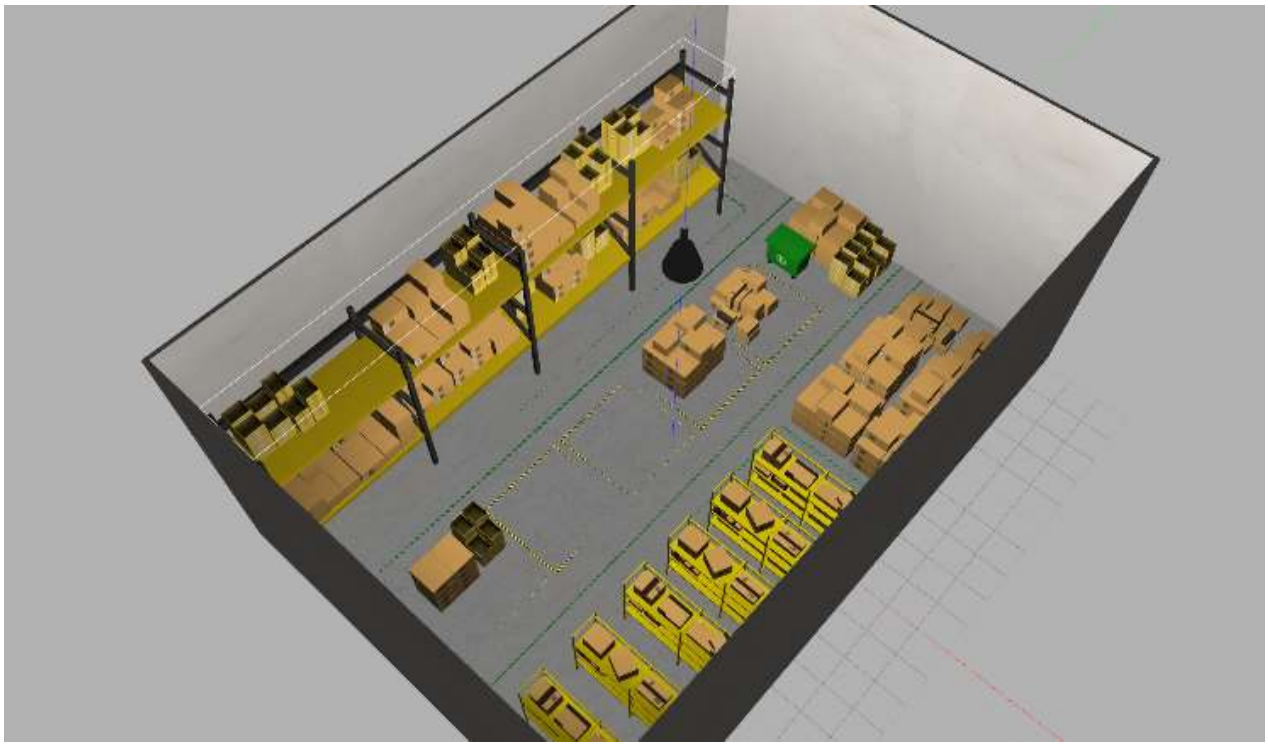


Рис. 4.3.1.1. Модель складского помещения AWS Robotics для симулятора Gazebo.

4.3.2. МОДЕЛИРОВАНИЕ СКЛАДСКОГО ПОМЕЩЕНИЯ ПО СТАНДАРТАМ ГОСТ В СИМУЛЯТОРЕ GAZEBO

Созданы две схемы складских помещений по стандартам ГОСТ (рис. 4.3.2.3, рис. 4.3.2.4). Склад №1 имеет размеры 16,5х50,5 м и высоту 8 м, что соответствует складскому помещению стандарта класса В. Он содержит 28 стеллажей размером 5700х1350х6000 мм (схема представлена на рис. 4.3.2.2). Расстояние от стены складского помещения до стеллажа 0,7 м, ширина коридора 3 м. Склад №2 имеет размеры 29х32,5 м и высоту 8 м, что также соответствует стандарту класса В. В складском помещении 8 стеллажей размером 28100х1350х6000 мм. Расстояние от стены складского помещения до стеллажа 0,7 м, ширина коридора 3,7 м.

Созданные модели стеллажей и складских помещений показаны на рис. 4.3.2.1, 4.3.2.5.

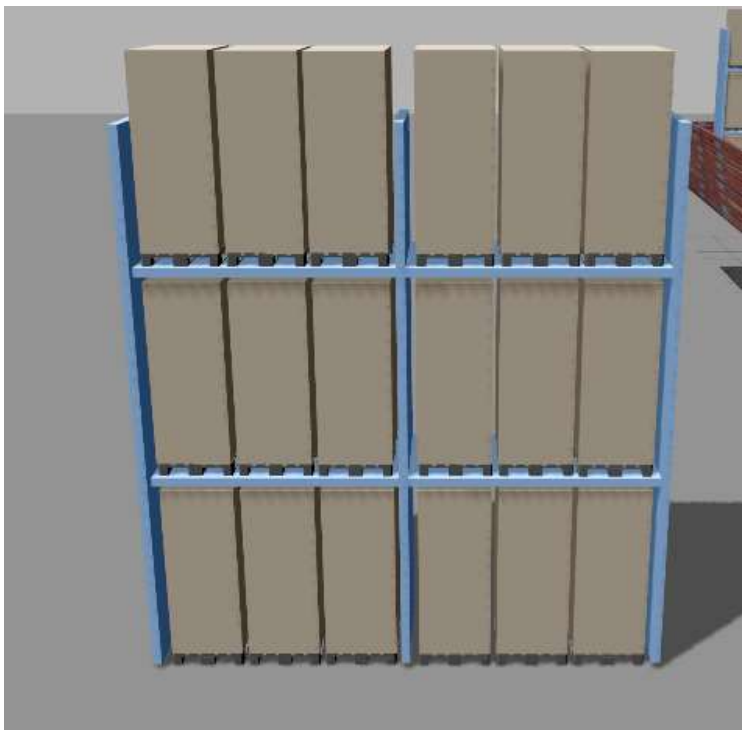


Рис. 4.3.2.1. Модель стеллажа для симулятора Gazebo.

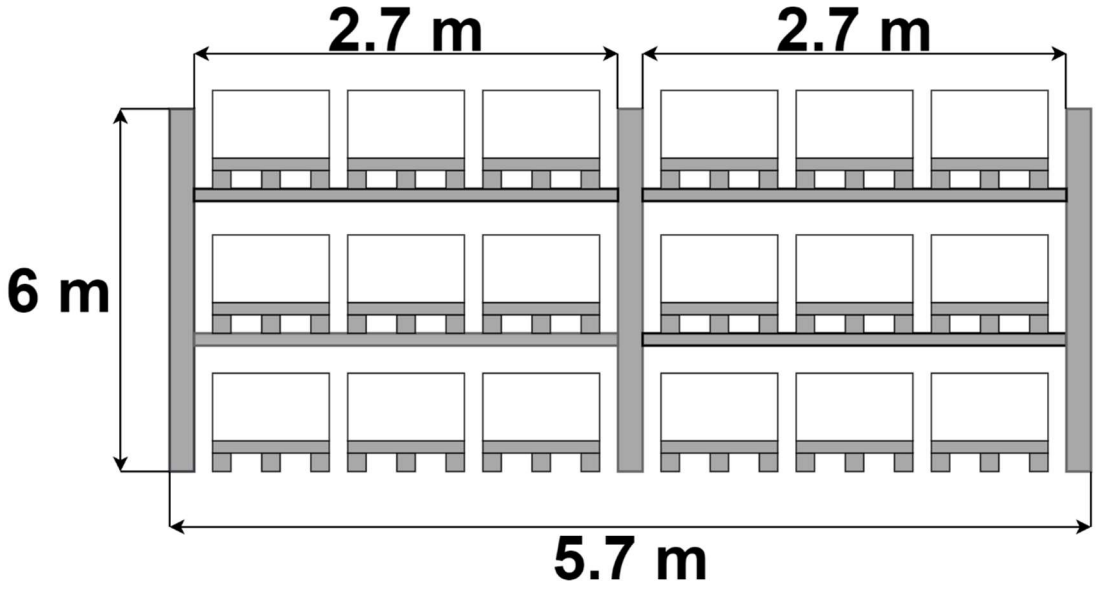


Рис. 4.3.2.2. Расположение товаров и паллетов на стеллаже.

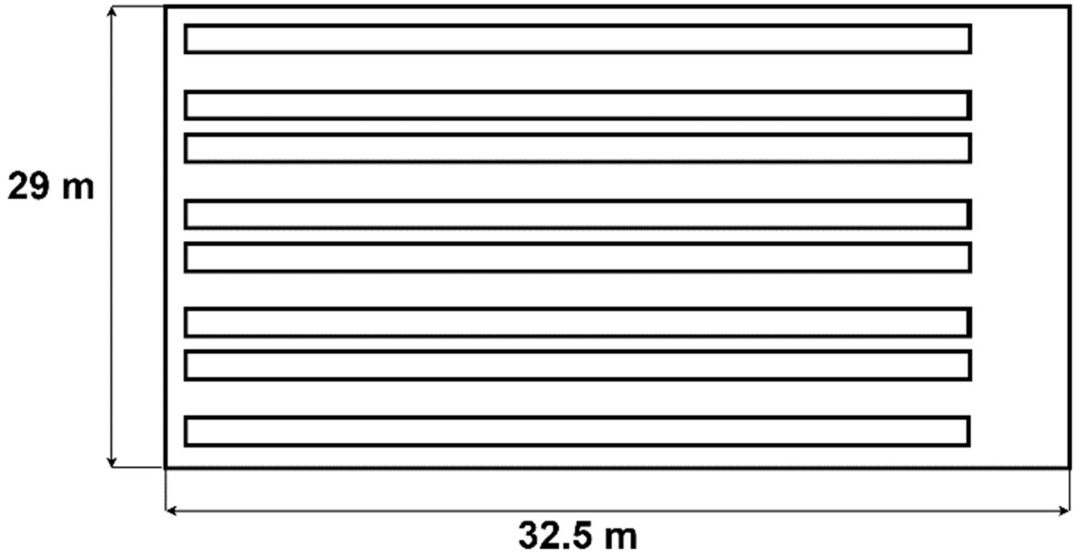


Рис. 4.3.2.3. Схема складского помещения размеров 29х32.5 м.

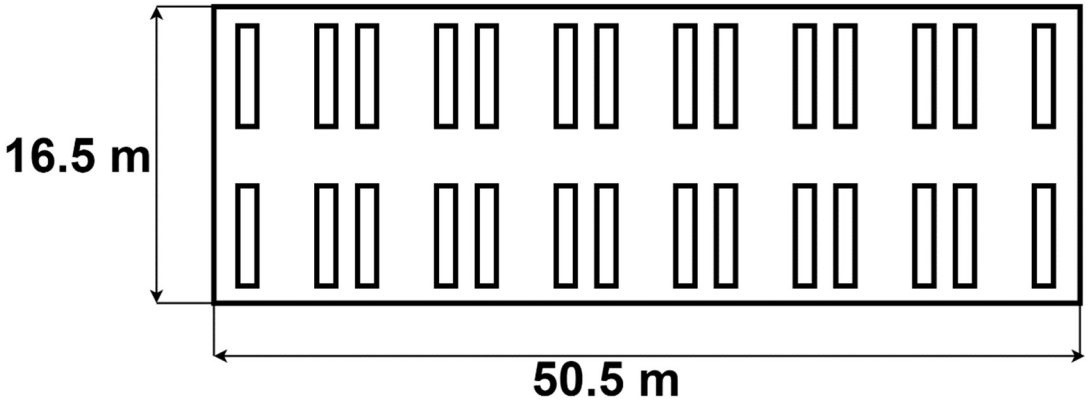
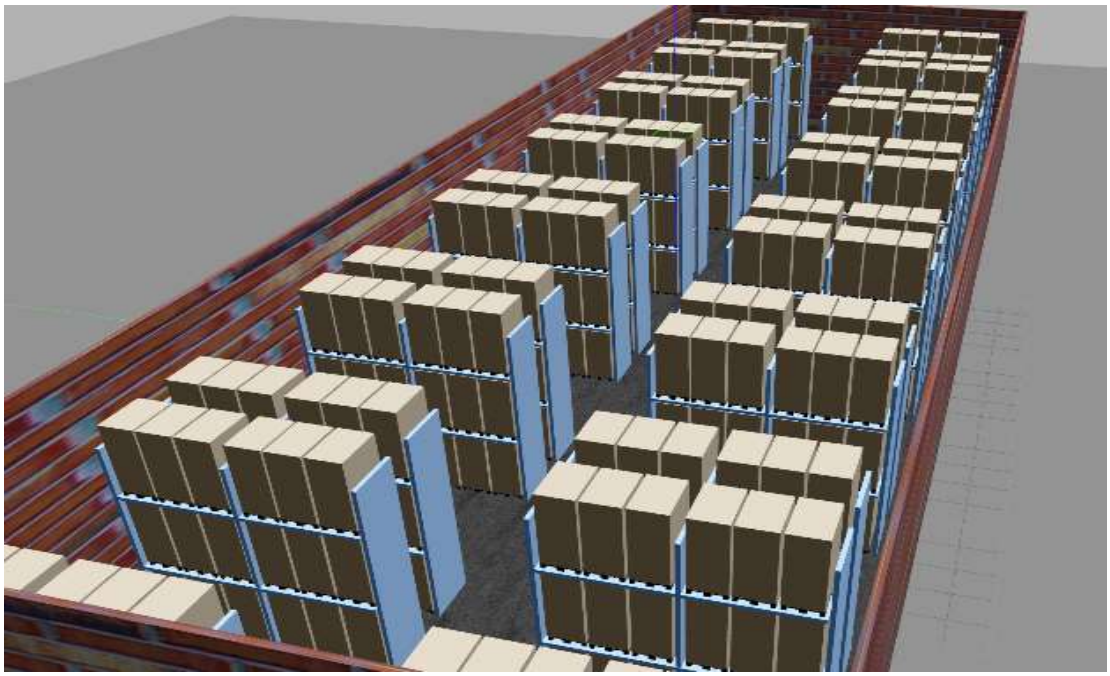


Рис. 4.3.2.4. Схема складского помещения размеров 16.5x50.5 м.

Разработанные модели складского помещения могут использоваться вместе с координатными метками и RFID-системами [40] для моделирования задач инвентаризации и навигации в среде ROS / Gazebo.



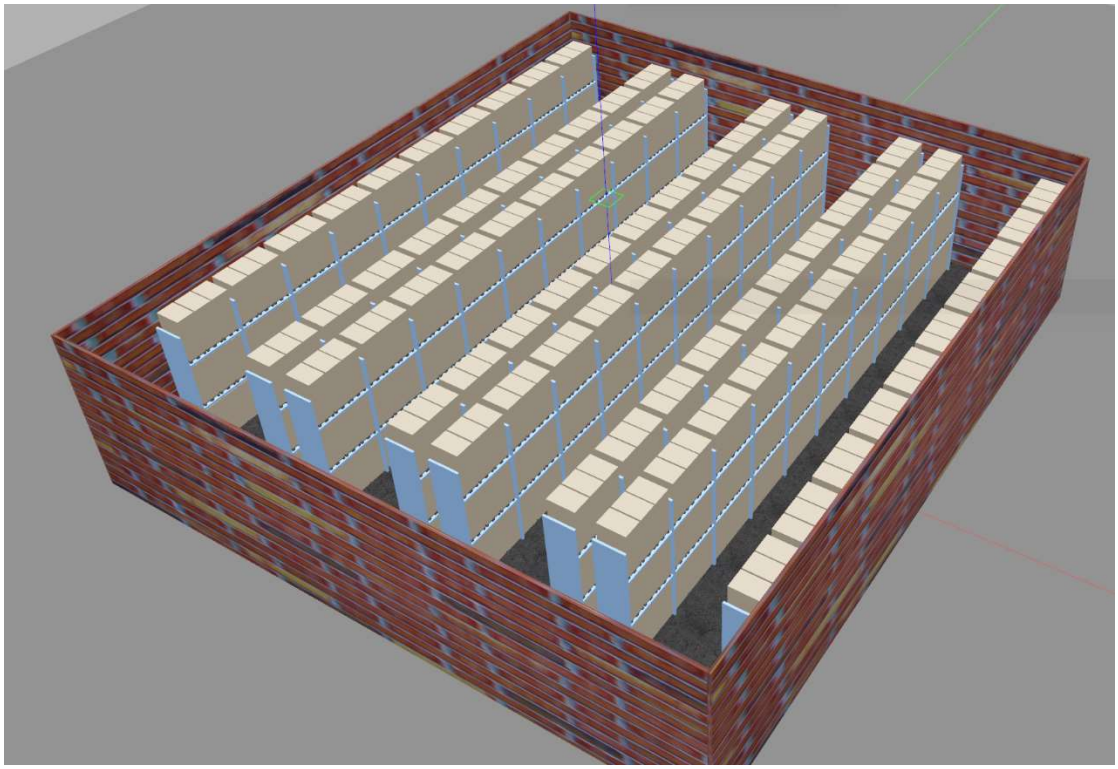


Рис.4.3.2.5. Модели складского помещения в симуляторе Gazebo. Складское помещение размером 16.5x50.5 м (наверху). Складское помещение размером 29x32.5 м (внизу).

Автономная навигация может выполняться с использованием координатных меток, прикрепленных к стеллажам складского помещения. Такой метод навигации можно восстановить после сбоя локализации, поэтому он более надежен, чем существующие коммерческие решения.

Модели складских помещений отражают расположение грузов на реальных складах и может быть использовано для проверки алгоритмов автономной инвентаризации. Размеры модели стеллажа можно редактировать, используя SDF файл модели. Модели грузов, расположенные на модели стеллажа, также можно редактировать или добавлять свои.

4.3.3. ОЦЕНКА ПРОИЗВОДИТЕЛЬНОСТИ ВИРТУАЛЬНОЙ МОДЕЛИ

Все модели складских помещений, описанные в этой работе, были оценены в симуляторе Gazebo с роботами TurtleBot 3 и TIAGo Base. Эти роботы оснащены различными датчиками, такими как лидары, визуальные камеры, датчики одометрии и другие. Следовательно, их использование требует правильного физического моделирования. Это загружает центральный процессор рабочей станции и имитирует реальное использование созданных сред.

Для оценки загрузки центрального процессора использовались показатели Real Time Factor (RTF). Эти метрики встроены в симулятор Gazebo и показывают соотношение между смоделированным временем и реальным временем, необходимым для расчета моделирования. Например, если

значение RTF равно 0,5, это означает, что для обработки одной секунды моделирования требуется две секунды реального времени. Для каждого смоделированного складского помещения была построена его карта с использованием алгоритма gmapping [41] (рис.4.3.3.1). Минимальные значения RTF во время сопоставления представлены в сравнительной таблице 4.3.3.1.

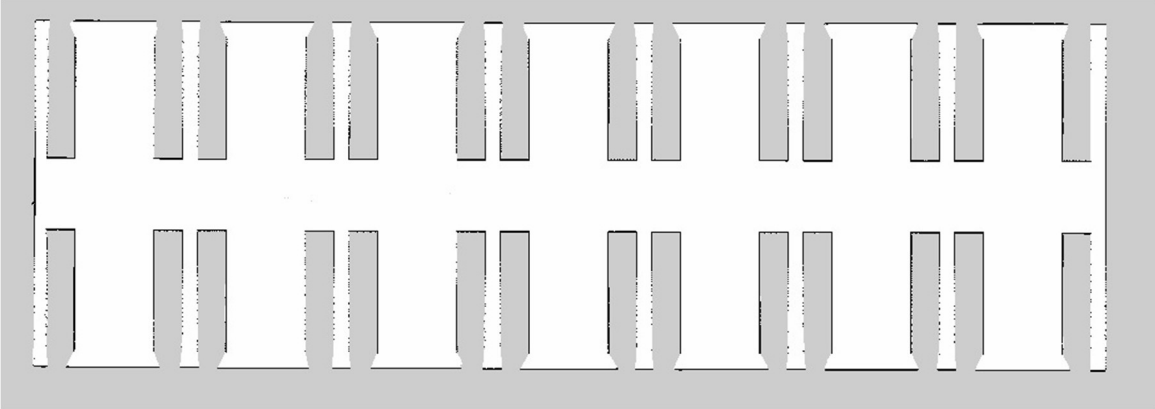


Рис. 4.3.3.1. Карта местности модели складского помещения 16.5x50.5 м.

Таблица 4.3.3.1. Минимальное значение RTF в зависимости от выбранной модели робота и складского помещения.

Модель робота	Модель складского помещения		
	Складское помещение 16.5x50.5 м	Складское помещение 29x32.5 м	Складское помещение AWS Robotics, 14x21 м
TurtleBot 3	0.97	0.93	0.98
TIAGo Base	0.77	0.70	0.95

Как видно, модель складского помещения от AWS Robotics имеет лучший RTF в случае обоих роботов. Разработанные модели показывают

более низкий RTF; однако они имеют значительно большую площадь и большее количество расположенных объектов. Более того, RTF не уменьшается и остается стабильным.

5. СОЗДАНИЕ СИСТЕМЫ АВТОМАТИЗАЦИИ ЗАДАЧ ИНВЕНТАРИЗАЦИИ СКЛАДСКОГО ПОМЕЩЕНИЯ С ИСПОЛЬЗОВАНИЕМ ЛОКАЛИЗАЦИИ НАЗЕМНОГО РОБОТА-ДОСТАВЩИКА ПРИ ПОМОЩИ ARUCO-МАРКЕРОВ

Создание системы автоматизации задач инвентаризации складского помещения с использованием локализации наземного робота-доставщика при помощи ArUco-маркеров состоит из нескольких этапов:

1) Создание RFID-системы для использования в среде ROS/Gazebo
2) Создание программы для посадки БЛА PX4-LIRS на ArUco-маркер
3) Навигация мобильного робота TIAGo Base по путевым точкам в складском помещении
4) Локализация мобильного робота TIAGo Base относительно обнаруженного ArUco-маркера
5) Выравнивание мобильного робота TIAGo Base относительно обнаруженного ArUco-маркера
6) Создание контроллера для управления созданной системы

5.1. СОЗДАНИЕ RFID-СИСТЕМЫ ДЛЯ ИСПОЛЬЗОВАНИЯ В СРЕДЕ ROS/GAZEBO

Радиочастотная идентификация (RFID) — это использование радиоволн для чтения и записи информации, хранящейся в RFID-метке, прикрепленной к объекту. RFID-метка может быть прочитана на расстоянии до нескольких

метров и не обязательно должна находиться в пределах прямой видимости считывающего устройства, чтобы ее можно было отследить.

Система RFID состоит из двух частей: метки и считывателя. RFID-метки или метки встроены в передатчик и приемник. Компонент RFID на метках состоит из двух частей: микрочипа, который хранит и обрабатывает информацию, и антенны для приема и передачи сигнала. RFID-метка содержит конкретный серийный номер для одного конкретного объекта. Чтобы прочитать информацию, закодированную в метке, используют RFID-считыватель, он излучает сигнал на метку с помощью антенны. RFID-метка отправляет информацию, которая хранится в ее памяти. Затем RFID-считыватель передает результаты считывания в компьютерную программу RFID.

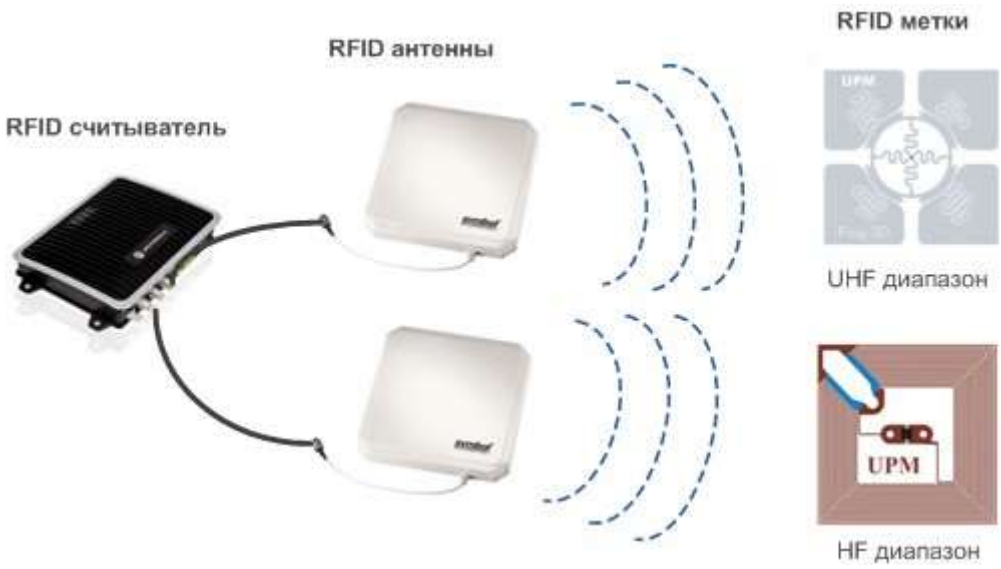


Рис. 5.1.1. Метод работы RFID-системы[8].

Есть три типа RFID-меток: пассивные, активные и полуактивные. Пассивная RFID-метка будет использовать энергию радиоволны считывателя для передачи сохраненной информации обратно на

[8] Фото взято с https://posmagazin.ru/userfiles/clauses/large/13/123071_iz-chego-sostoit-sistema-rfid.jpg

считыватель. Активная RFID-метка использует аккумулятор, который питает реле информации. Полуактивная RFID-метка оснащена батареей, которая обеспечивает чип электропитанием. При этом дальность считывания полуактивных RFID-меток зависит только от чувствительности RFID-считывателя; в отличии от полуактивных RFID-меток, дальность считывания активных меток зависит не только от чувствительности RFID-считывателя, но и от чувствительности самой метки. В розничной торговле RFID-метки могут быть прикреплены к различным предметам. Когда сотрудник использует портативный RFID-считыватель для сканирования полок, он может различать две пары идентичных товаров на основе информации, хранящейся на RFID-метке. У каждой пары будет свой серийный номер. RFID-метки по частоте работы делятся на следующие типы:

1) Low frequency (LF)
2) High frequency (HF)
3) Ultra-High frequency (UHF)

Чем выше частота RFID-метки, тем выше дальность считывание данной RFID-метки. В таблице указаны частоты, на которых могут работать RFID-метки.

Таблица 5.1.1. Сравнение частоты и дальности считывания

Частота	Дальность считывания
LF (125 - 134 кГЦ)	До 10 см
HF (13, 56 МГц)	До 30 см
UHF (860 - 960 МГц)	Больше 100 см

ALIEN 9640

Рис. 5.1.2. Пассивная RFID-метка ALIEN 9640[9].

Рис.5.1.3. Мобильный RFID-считыватель ThingMagic[10] [42].

В данной работе была использована RFID-система, которую я создал во время написания своей дипломной работы во время учебы в бакалавриате [43]. RFID-система, разработанная для использования в трехмерном динамическом симуляторе Gazebo, состоит из беспроводного передатчика и беспроводного приемника.

[9] Фото взято с https://lh3.googleusercontent.com/proxy/tn1F6wmPTTjn3rpinhN-ciotdOLwzJKKSh3TyWVOY38JRY5WetvAgWGoO-Fz2osxPG1kRWr3fkwPEiUOZSKtTJTW90t3Mbw4

[10] Фото взято с https://cdn11.bigcommerce.com/s-ka7ofex/images/stencil/1280x1280/products/1893/9587/ThingMagic_USB_Pro_RFID_Reader_Front.a76ef047d8d24c03823acdf41c4ee7c8__47986.1542759743.jpg?c=2

Беспроводной приемник используется в качестве RFID-считывателя, а беспроводной передатчик используется в качестве RFID-метки. При попадании сигнала RFID-метки в радиус считывания RFID-считывателя, RFID-метка отправляет данные, которые хранятся на RFID-метке, на RFID-считыватель. Беспроводной передатчик имеет следующие поля:

1) ESSID – Имя или ID
2) Frequency – частота, которую использует беспроводной передатчик
3) Signal level – уровень сигнала беспроводного передатчика
4) Sensitivity – чувствительность беспроводного передатчика

Дальность считывания RFID-метки зависит от уровня чувствительности, чем она выше, тем дальше дальность считывания. Также была создана программа, которая записывает данные обнаруженных RFID-меток в соответствующий файл (рис.5.1.4).

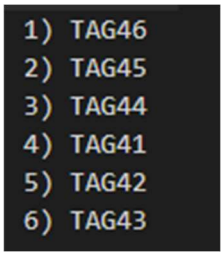


Рис. 5.1.4. Данные обнаруженных RFID-меток.

Данная RFID-система предназначена для решения задач, связанных с инвентаризацией в складских помещениях.

5.2. СОЗДАНИЕ ПРОГРАММЫ ДЛЯ ПОСАДКИ БЛА PX4-LIRS НА ARUCO-МАРКЕР

Для инвентаризации товаров, оборудованных RFID-метками, был использован БЛА PX4-LIRS оборудованный RFID-считывателем, разработанным в разделе 5.1. Основной задачей БЛА PX4-LIRS является считывание обнаруженных RFID-меток в складском помещении. Для перемещения БЛА в складском помещении был использован мобильный робот TIAGo Base. БЛА оборудован камерой, которая позволяет позиционироваться относительно обнаруженного ArUco-маркера, установленного на поверхность мобильного робота TIAGo Base (рис.3.3.2). Полет БЛА в помещении на основе данных, полученных с GPS, невозможен, т.к. сигнал GPS в помещении будет либо очень слабым, либо сигнала не будет вообще. Для решения данной проблемы БЛА использовал камеру оптического потока PX4FLOW и ультразвуковой датчик MB1242.

5.2.1. ДОПОЛНИТЕЛЬНЫЕ СЕНСОРЫ БЛА PX4-LIRS

PX4Flow (рис.5.2.1.1) [44] — это интеллектуальная камера с оптическим потоком. PX4Flow имеет собственное разрешение 752 × 480 пикселей и рассчитывает оптический поток на 4-кратном биннинге и кадрировании области с частотой 250 Гц, что обеспечивает очень высокую светочувствительность. В отличие от многих аналогичных датчиков, он также работает в помещении и в условиях низкой внешней освещенности без необходимости использования светодиода с частотой 120 Гц. Его можно перепрограммировать для выполнения любой другой базовой, эффективной низкоуровневой задачи компьютерного зрения.

Технические характеристики:

1) Процессор: Cortex M4F с тактовой частотой 168 МГц (128 + 64 КБ ОЗУ)
2) Датчик изображения: MT9V034 752×480,
3) Объектив: L3GD20 3D Gyro 16 мм M12

PX4FLOW используется для удержания позиции БЛА в местах с ограниченным сигналом GPS.

Рис.5.2.1.1. Камера оптического потока PX4Flow[11].

MB1242 (рис.5.2.1.2) [45] - ультразвуковой датчик I2XL-MaxSonar-EZ4 с наибольшей помехоустойчивостью и самой узкой зоной обнаружения из всех датчиков I2XL-MaxSonar-EZ. Датчик разработан и откалиброван для обеспечения надежной информации о расстоянии до крупных целей даже в средах с сильными акустическими или электрическими источниками шума. Устойчив к внешним акустическим и электромагнитным помехам.

Отличительные особенности:

1) Напряжение питания +3...+5,5 В

[11] Фото взято с https://lh3.googleusercontent.com/proxy/Rh50T-SAqIiTwwsmTLNdGYN-JSz9LsG5xuwYkwhHo5tGIGOsZntNMLMTpqrYFryeGwsDrAEivkeSQqVjXteJ-oaR1Yy56tWvqd2ECQ

2) средний ток потребления: 3,4 мА
3) рабочая частота 42 кГц
4) интервал считывания показаний 67 мс (15 Гц)
5) максимальное рабочее расстояние: 765 см
6) разрешение в диапазоне от 25 до 765 см: 1 см
7) интерфейс I2C
8) рабочая температура 0...+65 °C

Основные области применения: обнаружение людей, робототехника, измерение расстояния, системы безопасности, квадрокоптеры, беспилотные летающие аппараты. Ультразвуковой датчик MB1242 используется для удержания высоты БЛА в помещении.

Рис. 5.2.1.2. Ультразвуковой датчик MB1242[12].

5.2.2 ARUCO-МАРКЕРЫ

Оценка позы имеет большое значение во многих приложениях компьютерного зрения: навигации роботов, дополненной реальности и многих других. Этот процесс основан на нахождении соответствий между точками в реальной среде и проекцией их 2-мерного изображения. Нахождение соответствий между точками в реальной среде и проекцией их 2-мерного

[12] Фото взято с https://www.maxbotix.com/wp-content/uploads/2017/01/XL-Ultrasonic-Sensor-Iso-768x768.jpg

изображения трудный шаг, поэтому для облегчения обычно используют синтетические или координатные метки.

Один из самых популярных подходов - использование двоичных квадратных меток координат [46]. Основное преимущество этих маркеров заключается в том, что один маркер обеспечивает достаточное количество соответствий (его четыре угла) для получения позы камеры. Кроме того, внутренняя двоичная кодификация делает их особенно надежными, позволяя применять методы обнаружения и исправления ошибок.

ArUco-маркер - это синтетический квадратный маркер, состоящий из широкой черной границы и внутренней двоичной матрицы, которая определяет его идентификатор (id). Черная рамка способствует ее быстрому обнаружению на изображении, а двоичная кодификация позволяет ее идентифицировать и применять методы обнаружения и исправления ошибок. Размер маркера определяет размер внутренней матрицы.

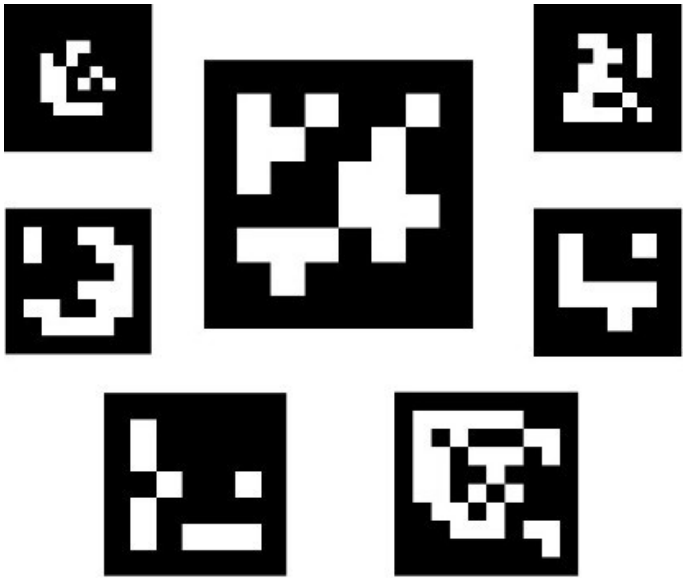


Рис. 5.2.2.1. ArUco-маркеры[13].

[13] Фото взято с https://docs.opencv.org/3.4/markers.jpg

Следует отметить, что маркер можно найти повернутым в окружающей среде, однако процесс обнаружения должен иметь возможность определять его исходное вращение, чтобы каждый угол идентифицировался однозначно. Это также делается на основе двоичной кодификации.

Словарь маркеров — это набор маркеров, которые рассматриваются в конкретном приложении. Это просто список двоичных кодификаций каждого из его маркеров. Основными свойствами словаря являются размер словаря и размер маркера:

- Размер словаря — это количество маркеров, составляющих словарь.
- Размер маркера — это количество бит, которое содержит маркер.

Модуль ArUco включает несколько предопределенных словарей, охватывающих диапазон различных размеров словарей и размеров маркеров.

Идентификатор маркера — это число, полученное в результате преобразования двоичной кодификации в десятичное базовое число. Однако это невозможно, поскольку для маркеров большого размера количество битов слишком велико, и управление такими огромными числами нецелесообразно. Вместо этого идентификатор маркера — это просто индекс маркера в словаре, которому он принадлежит. Например, первые 5 маркеров в словаре имеют идентификаторы: 0, 1, 2, 3 и 4.

Учитывая изображение, содержащее маркеры ArUco, процесс обнаружения должен вернуть список обнаруженных маркеров. Каждый обнаруженный маркер включает:

- Положение его четырех углов на изображении (в исходном порядке) (рис. 5.5.2.2).
- Идентификатор маркера.

Процесс обнаружения маркеров состоит из двух основных этапов:

1. Обнаружение кандидатов в маркеры. На этом этапе изображение анализируется, чтобы найти квадратные формы, которые могут быть маркерами. Он начинается с адаптивного определения порога для сегментации маркеров, затем контуры извлекаются из изображения с пороговым значением, а те, которые не являются выпуклыми или не приближаются к квадратной форме, отбрасываются. Также применяется некоторая дополнительная фильтрация (удаление слишком маленьких или слишком больших контуров и т.д.).
2. После обнаружения кандидатов необходимо определить, действительно ли они являются маркерами, путем анализа их внутренней кодификации. Этот шаг начинается с извлечения маркерных битов каждого маркера. Для этого сначала применяется перспективное преобразование, чтобы получить маркер в его канонической форме. Затем каноническое изображение определяется с помощью метода Оцу [47] для разделения белых и черных битов. Метод Оцу - это алгоритм вычисления порога бинаризации для полутонового изображения, используемый в области компьютерного распознавания образов и обработки изображений для получения чёрно-белых изображений. Изображение разделено на разные ячейки в соответствии с размером маркера и размером границы. Затем подсчитывается количество черных или белых пикселей в каждой ячейке, чтобы определить, является ли это белым или черным битом. Наконец, биты анализируются, чтобы определить, принадлежит ли маркер определенному словарю. При необходимости используются методы исправления ошибок.

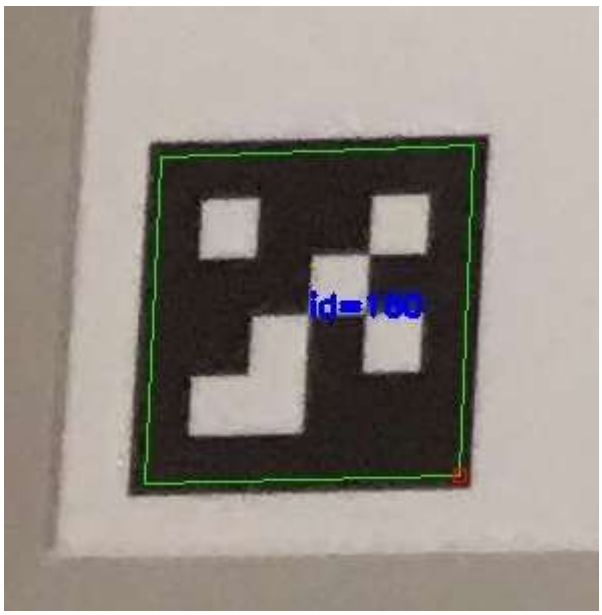


Рис. 5.2.2.2. Обнаружение ArUco-маркера.

5.2.3 ВЛОЖЕННЫЙ ARUCO: НОВЫЙ ПОДХОД ДЛЯ ПОСАДКИ БЛА С ВЫСОКОЙ ТОЧНОСТЬЮ

Современные БЛА активно используются в различных сферах как в военных, так и в гражданских целях. К типичным задачам относятся поиск различных объектов на больших площадях [48], мониторинг и контроль окружающей среды [49], проверка и мониторинг конструкций [50], доставка товаров [51] и другие [52]. Все эти задачи требуют, чтобы БЛА взлетал в начале операции и приземлялся после выполнения своей задачи. Однако точная посадка без использования систем глобального позиционирования (например, в помещении) часто требует значительного вмешательства человека для обеспечения успешной посадки. Это становится особенно важным для критических миссий и в ситуациях, когда БЛА имеет на борту дорогостоящее оборудование, поскольку большинство аварий БЛА происходит во время процедуры посадки и вызвано неожиданной жесткой посадкой [53].

Таким образом, системы помощи при посадке — это обычный способ уменьшить количество инцидентов. Проблема точной посадки может быть решена путем повышения точности оценки позы БЛА.

Оценка позы робота на основе визуальных сенсорных данных является ключевой функцией во многих робототехнических приложениях: локализация [54], навигация роботов [55], SLAM [56] и другие [57]. Этот процесс основан на нахождении соответствий между характерными точками в реальной среде и их проекцией на двумерное изображение. Этот процесс требует больших вычислительных ресурсов для произвольного изображения. Поэтому координатные метки обычно используются для выделения подходящих характерных точек. Предварительно определенные размер и пропорции координатной метки позволяет определить относительный поворот камеры и оценить расстояние до нее.

Одним из наиболее популярных подходов является использование бинарных квадратных меток координат. Внутреннее двоичное кодирование увеличивает надежность обнаружения с помощью методов автоматического обнаружения и исправления ошибок. В этой работе используются ArUco-маркеры [58], которые очень популярны в проектах дополненной реальности для оценки положения камеры [59].

В этом разделе представлен разработанный мною новый подход, направленный на решение проблемы точной посадки БЛА с использованием координатных меток. В нем используется модифицированный маркер ArUco, который был назван вложенный ArUco-маркер (embedded ArUco или e-ArUco). Маркеры e-ArUco могут быть обнаружены на протяжении всего процесса посадки, как с больших, так и с малых расстояний, и для обнаружения требуются только оригинальные алгоритмы обнаружения ArUco. Предложенные e-ArUco маркеры были протестированы с помощью виртуальных экспериментов в симуляторе Gazebo.

Пакет ROS aruco_detect обнаруживает маркеры ArUco на изображениях с камеры. Результаты обнаружения в дальнейшем используются полетным контроллером для определения направления движения БЛА. Установленный на борт БЛА на Pixhawk Raspberry Pi 4 использует Ubuntu Mate 18.04 и ROS Melodic. Трёхмерный динамический симулятор Gazebo на основе ROS был использован для проверки предложенного подхода e-ArUco и разработанного программного обеспечения в виртуальной среде. На модели БЛА была установлена направленная вниз монокулярная камера для обнаружения маркеров ArUco на посадочной поверхности под БЛА. Программное обеспечение БЛА также имитировалось, чтобы упростить перенос программного обеспечения из Gazebo на настоящий БЛА в будущем.

Новый маркер e-ArUco — это ArUco-маркер, который имеет дополнительный внутренний маркер в геометрическом центре исходного (внешнего) ArUco-маркера. Два маркера (внешний и внутренний) имеют разные идентификаторы. Основной особенностью предлагаемого решения является возможность надежного обнаружения маркера на протяжении всего процесса посадки дрона. Внешний маркер используется для приблизительного позиционирования БЛА на большой высоте, а внутренний маркер используется на поздних этапах приземления на малой высоте, когда внешний маркер находится слишком близко к камере и, таким образом, не полностью захватывается кадрами камеры видеопотока (поскольку область маркера становится слишком большой, и некоторая ее часть оказывается за пределами кадра). Эта функция повышает точность приземления, поскольку маркер e-ArUco фиксируется камерой на всем протяжении процедуры приземления. При выборе конкретного идентификатора [60] для внешнего маркера ArUco следует обратить внимание на то, что внутренний маркер ArUco заменяет один черный «пиксель» в его центре. Поэтому выбор внешнего идентификатора маркера ArUco ограничен теми, которые имеют черный блок в центре, в то

время как рекомендуется выбирать внутренний маркер ArUco таким образом, чтобы количество его черных пикселей преобладало над количеством белых.

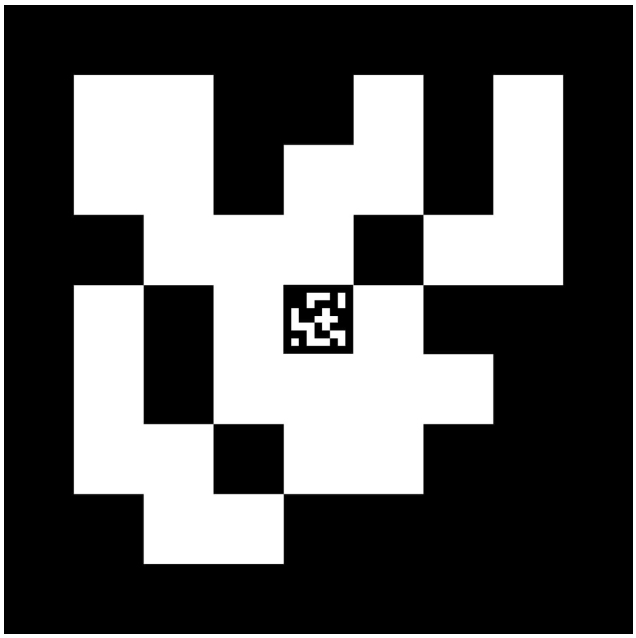


Рис. 5.2.3.1. e-ArUco маркер.

Экспериментальная установка, использованная в этой работе, состояла из двух маркеров ArUco размером 45x45 см для внешнего маркера и размером 5x5 см для внутреннего маркера. В таблице 5.2.3.1 описаны минимальные и максимальные расстояния, которые позволяют обнаруживать маркеры для внутренних маркеров ArUco, внешних маркеров ArUco и e-ArUco. Маркер e-ArUco обладает лучшими характеристиками, поскольку его можно обнаружить на расстоянии от 0,2 м до 30 м.

Таблица 5.2.3.1. Минимальные и максимальные диапазоны обнаружения для оригинальных маркеров ArUco и e-ArUco.

Тип маркера	Размер (см x см)	Расстояние обнаружения (м)	
		Минимальное	Максимальное
Внутренний ArUco	5x5	0.2	1.2
Внешний ArUco	45x45	0.8	30

e-ArUco	45x45	0.2	30

Пакет ROS aruco_detect [61] используется для обработки кадров камеры и извлечения маркеров e-ArUco. Этот пакет находит ArUco-маркеры в потоке изображений, публикует координаты их вершин (четырех угловых точек) и вычисляет положение камеры относительно маркера. Этот пакет поддерживает словари с разным количеством бит кодирования (от 4x4 до 7x7). Маркеры e-ArUco созданы на основе маркеров ArUco типа 7x7. Такой выбор позволяет разместить внутренний маркер непосредственно в центре внешнего маркера и не влияет на распознавание внешнего маркера. Зная относительное положение камеры относительно маркера, разработанное управляющее программное обеспечение способно точно перемещать БЛА и приземлять его на маркер e-ArUco. Алгоритм посадки работает следующим образом. БЛА получает команду на посадку во время полета. БЛА начинает поиск маркера e-ArUco. После успешного обнаружения БЛА выравнивается по центру внешнего ArUco и начинает снижаться. Во время процесса посадки БЛА постоянно корректирует свое положение, чтобы обнаруженный маркер оставался в центре кадра камеры. По мере приближения внешний маркер перестает попадать в поле зрения камеры, БЛА переключается на обнаружение внутреннего маркера и продолжает точную посадку, полагаясь на внутренний маркер.

Процесс посадки завершается, когда расстояние до e-ArUco маркера будет меньше 15 см.

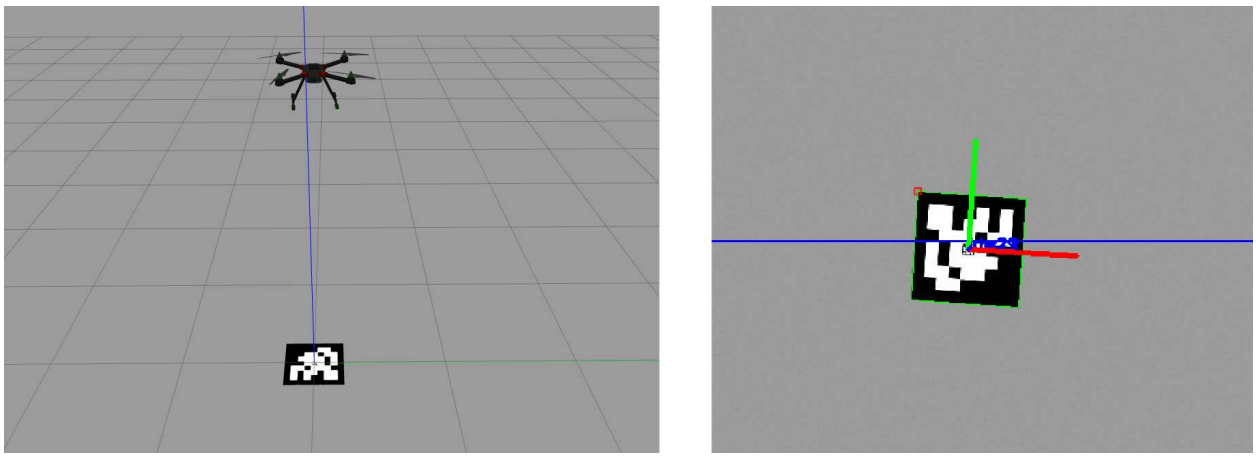


Рис. 5.2.3.2. Обнаружение внешнего маркера.

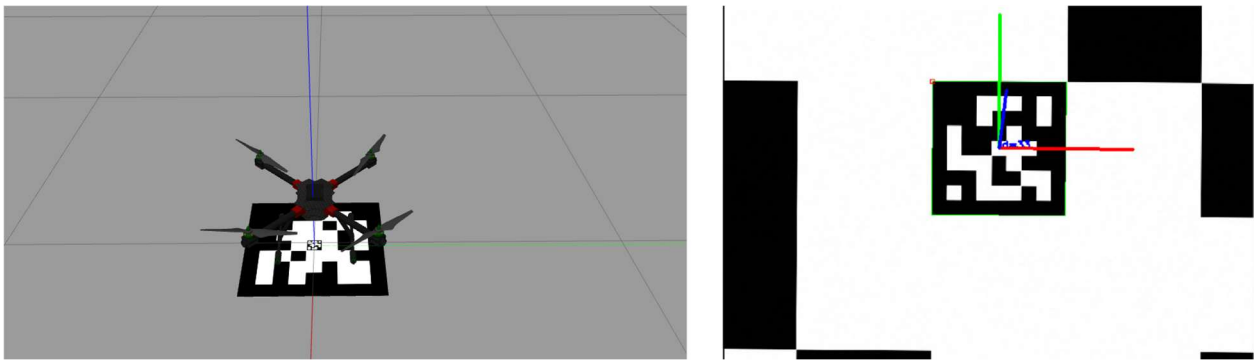


Рис. 5.2.3.3. Обнаружение внутреннего маркера.

Для оценки эффективности разработанной системы e-ArUco маркеров были проведены многочисленные эксперименты. Всего в симуляторе Gazebo было проведено 20 экспериментов. Во время испытаний БЛА взлетел на высоту 2 м и ждал команды посадки на e-ArUco маркер. После получения команды БЛА выполнил вышеупомянутую процедуру посадки. Когда посадка была завершена, было измерено расстояние между центром камеры БЛА и центром маркера e-ArUco. На рисунке 5.2.3.4 показаны результаты, полученные во всех испытаниях. Средняя точность составила 2,03 см при стандартном отклонении 1,53 см.

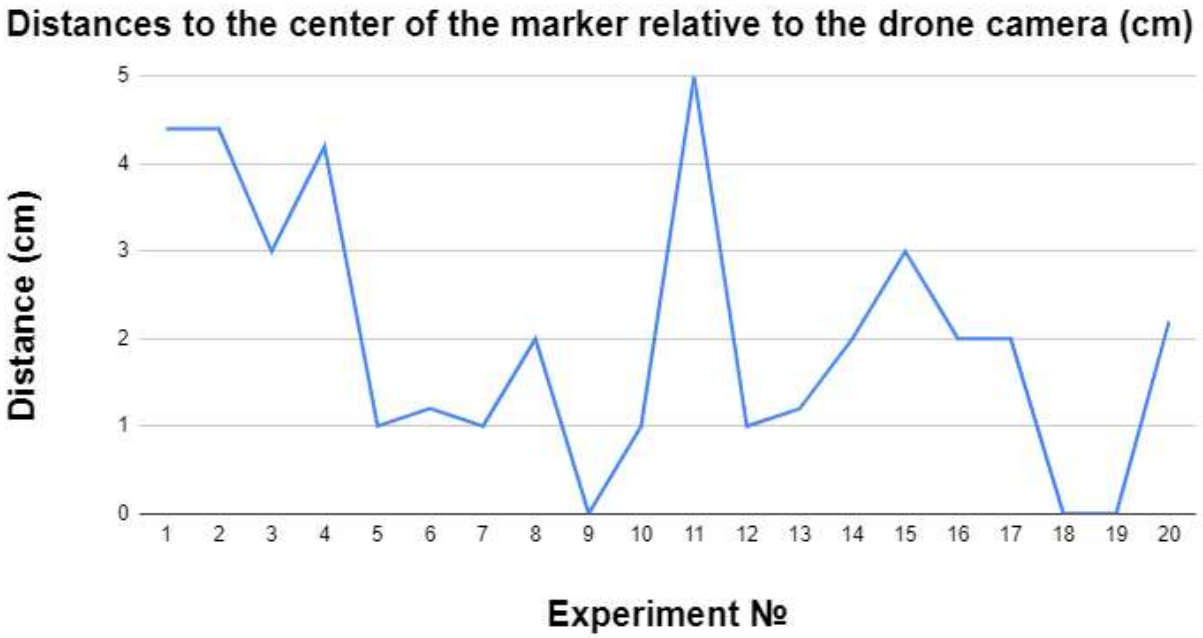


Рис. 5.2.3.4. Расстояние до центра маркера относительно центра камеры БЛА.

Дополнительно были проанализированы и изучены научные публикации по тематике точной посадки БЛА на ArUco-маркер и были выделены следующие работы:

- Авторы [62] представили два алгоритма. Первый алгоритм был сконцентрирован на обнаружении маркера ArUco в изображении камеры БЛА, что позволяет определять положение и выравнивание камеры БЛА относительно маркера. Второй алгоритм выполнял посадку БЛА непосредственно на поверхность маркера. В него входил программный модуль для корректировки положения БЛА относительно маркера с целью повышения точности посадки. Разработанные алгоритмы были протестированы в среде симулятора Gazebo.
- В [63] авторы также использовали маркер ArUco для высокоточной посадки БЛА. В предложенных решениях статического и динамического подхода БЛА был оснащен недорогой камерой Raspberry Pi и был способен обнаруживать ArUco-маркер размером 56 × 56 см с высоты до 30 м. Метод был проверен с использованием платформы ArduSim [20] и

экспериментов с реальным БЛА. Авторы сообщили о среднем смещении 11 см от положения цели и сравнили его с несколькими другими методами, подчеркнув довольно низкую точность традиционного приземления на основе GPS со смещением от 1 до 3 м.

- Авторы [64] использовали пару вложенных ArUco-маркеров для выполнения дневной и ночной посадки БЛА. Внешний маркер имел размер 70×70 см и содержал 8×8 двоичных ячеек: внутреннюю матрицу 6 × 6 (которая формирует идентификатор) и внешнюю границу черного цвета шириной в 1 ячейку. Внешний маркер размещался на белом фоне и мог быть обнаружен с расстояния около 17 м. В центре внешнего маркера авторы поместили внутренний маркер меньшего размера размером 12×12 см (маркер 8×8 ячеек, состоящий из внутренней матрицы 6×6 ячеек и границы черного цвета шириной в 1 ячейку), который можно было обнаружить на расстоянии около 5 м и позволял направлять процесс приземления, когда более крупный внешний маркер выходит из поля зрения камеры. Следует отметить, что на расстоянии от 1 до 2,5 м одновременно успешно обнаруживались оба маркера. Авторы подтвердили свой подход серией экспериментов с мультикоптером 3DRobotics X8.

Последний подход аналогичен нашему, но имеет ряд недостатков, которые мне удалось устранить. Основным недостатком решения является отсутствие корреляции между внешними и внутренними маркерами, что предотвращает умное встраивание, которое в дальнейшем может быть масштабируемым. Более того, поскольку внутренний маркер перекрывает четыре ячейки внешнего маркера, это уменьшает количество возможных вариантов выбора идентификаторов как внешних, так и внутренних маркеров, поскольку передний план (внутренний маркер) должен четко отличаться от четырех ячеек фона (внешний маркер).

5.3 АВТОНОМНАЯ НАВИГАЦИЯ МОБИЛЬНОГО РОБОТА TIAGO BASE ПО ПУТЕВЫМ ТОЧКАМ В СКЛАДСКОМ ПОМЕЩЕНИИ

Мобильный робот TIAGo Base имеет готовые ROS пакеты для автономной навигации и локализации. Для автономной навигации мобильного робота TIAGo Base в складском помещении, необходимо построить карту местности. ROS пакет pmb2_2dnav_gazebo содержит все необходимые данные для построения карты местности складского помещения. При помощи данного пакета можно построить карту местности используя данные лидара. Для построения карты местности использовался пакет gmapping. Пакет gmapping обеспечивает одновременную локализацию и картографирование на основе лидара в виде узла ROS, называемого slam_gmapping. Используя slam_gmapping, можно создать двухмерную карту местности (например, план этажа) с помощью лидара и данных позиции, собранные мобильным роботом.

Для разработки автономной навигации по путевым точкам был использован стек actionlib [65] и пакет move_base [66]. Стек actionlib предоставляет собой стандартизированный интерфейс для взаимодействия с приоритетными задачами. Примеры этого включают перемещение робота в заранее определенное место, выполнение лазерного сканирования и возврат полученного облака точек. Пакет move_base предоставляет собой реализацию, которая, учитывая заданную цель в мире, будет пытаться достичь ее с помощью мобильного робота. Узел move_base связывает глобальный и локальный планировщики для выполнения своей задачи глобальной навигации. Он поддерживает любой глобальный планировщик, придерживающийся интерфейса nav_core::BaseGlobalPlanner, указанный в пакете nav_core, и любой локальный планировщик, придерживающийся интерфейса nav_core::BaseLocalPlanner, указанного в пакете nav_core. Узел move_base также поддерживает две карты затрат, одну для глобального

планировщика и одну для локального планировщика, которые используются для выполнения задач навигации.

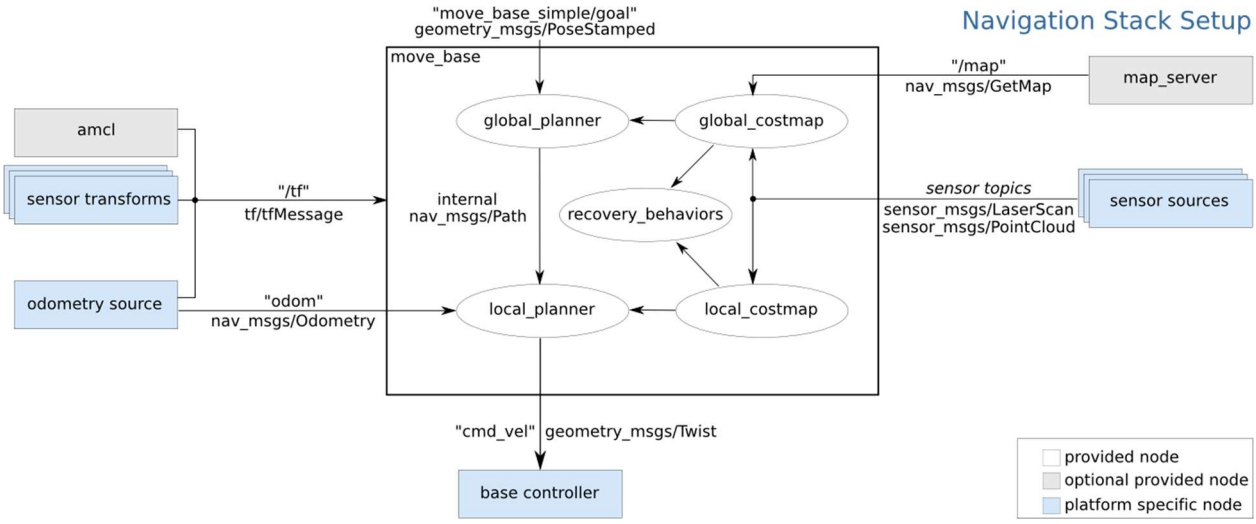


Рис.5.3.1. Схема работы пакета move_base[14].

Была разработана программа, которая отправляет путевые точки в move_base server. Данные позиции и ориентации цели хранятся в файле (рис. 5.3.2). Данные состоят из id цели, позиции цели и ориентации цели. Move_base server получает определенный id цели и отправляет данные позиции и ориентации заданной цели в стек навигации. При достижении заданной цели происходит отправка следующей цели; как только все целевые точки будут достигнуты, программа завершит работу.

[14] Фото взято с https://www.researchgate.net/profile/Leo-Ordinez/publication/271471748/figure/fig10/AS:668863683387395@1536480937098/Figura-10-Paquete-ROS-move-base-11.ppm

```
0 -16.4 -2.8 0.4 0 1.57 0
1 -16.4 -5.6 0.4 0 1.57 0
2 -15.6 -5.6 0.4 0 -1.57 0
3 -15.6 -2.8 0.4 0 -1.57 0
4 -9.3 -2.8 0.4 0 1.57 0
5 -9.3 -5.6 0.4 0 1.57 0
6 -8.5 -5.6 0.4 0 -1.57 0
7 -8.5 -2.8 0.4 0 -1.57 0
8 -2.2 -2.8 0.4 0 1.57 0
9 -2.2 -5.6 0.4 0 1.57 0
10 -1.4 -5.6 0.4 0 -1.57 0
11 -1.4 -2.8 0.4 0 -1.57 0
12 4.9 -2.8 0.4 0 1.57 0
13 4.9 -5.6 0.4 0 1.57 0
```

Рис.5.3.2. Список путевых точек.

5.4 ЛОКАЛИЗАЦИЯ МОБИЛЬНОГО РОБОТА TIAGO BASE ОТНОСИТЕЛЬНО ОБНАРУЖЕННОГО ARUCO-МАРКЕРА

Мобильный робот TIAGo Base использует ROS-пакет AMCL для локализации на известной карте. На концептуальном уровне пакет AMCL поддерживает распределение вероятностей по набору всех возможных поз робота и обновляет это распределение, используя данные одометрии и лидаров. Камеры глубины также можно использовать для создания этих двумерных лазерных сканирований с помощью пакета depthimage_to_laserscan [67], который принимает данные камеры глубины и публикует результаты лазерного сканирования sensor_msgs/LaserScan.

Пакет также требует предопределенной карты местности, с которой можно сравнивать наблюдаемые значения датчиков. На уровне реализации пакет AMCL представляет распределение вероятностей с использованием

фильтра частиц [68]. Фильтр является «адаптивным», потому что он динамически регулирует количество частиц в фильтре: когда поза робота неопределенна, количество частиц увеличивается; когда поза робота четко определена, количество частиц уменьшается. Это позволяет роботу находить компромисс между скоростью обработки и точностью локализации.

Несмотря на то, что пакет AMCL отлично работает из коробки, существуют различные параметры, которые можно настроить, основываясь на своих знаниях об используемой платформе и датчиках. Настройка этих параметров может повысить производительность и точность пакета AMCL и уменьшить количество оборотов для восстановления, которые робот выполняет во время навигации.

Существует три категории параметров ROS, которые можно использовать для настройки узла AMCL: общий фильтр, модель лидара и модель одометрии. Их можно редактировать в файле amcl.launch.

Использование ArUco-маркеров, как дополнительного источника данных, позволит повысить точность локализации мобильного робота TIAGo Base при их обнаружении. Мобильный робот TIAGo Base был оборудован камерой для обнаружения ArUco-маркера. ROS пакет aruco_detect использовался для работы с обнаруженными ArUco-маркерами. Используя полученные данные трансформации ArUco-маркера, можно определить на каком расстоянии находится центр камеры робота относительно центра обнаруженного ArUco-маркера.

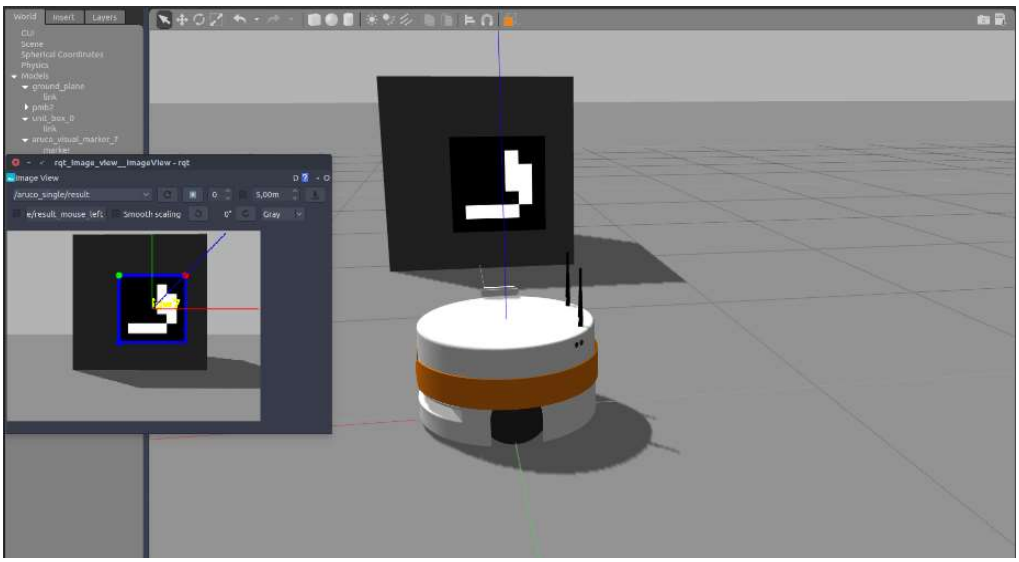


Рис. 5.4.1. Обнаружение ArUco-маркера мобильным роботом TIAGo Base.

Для локализации мобильного робота TIAGo Base используются данные позиции, ориентации ArUco-маркера и данные трансформации обнаруженного ArUco-маркера. Был создан файл, который содержит список всех ArUco-маркеров, их позицию и ориентацию. Для вычисления позиции мобильного робота TIAGo Base использовались данные обнаруженного ArUco-маркера и данные одометрии. Для фильтрации полученных данных использовался упрощенный фильтр Калмана [69]. Фильтр Калмана — эффективный рекурсивный фильтр, оценивающий вектор состояния динамической системы, используя ряд неполных и зашумленных измерений. Фильтр Калмана дает право воспринимать слишком быстрые скачки данных, как ошибку измерения. Он позволит получить более точную позицию робота используя информацию с различных источников, таких как данные ArUco-маркера и данные одометрии.

Формула упрощенного фильтра Калмана:

$M_n = kAn + (1 - k)M_{n-1}$, где

M_n - результирующее значение текущего вычисления,

k - коэффициент стабилизации,

A_n - исходное значение текущего измерения,

M_{n-1} - результирующее значение предыдущего вычисления.

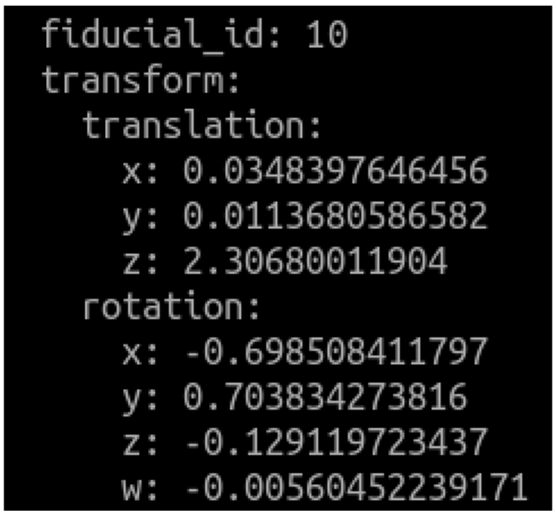


Рис. 5.4.2. Данные трансформации обнаруженного ArUco-маркера.

Для визуализации возможных позиций мобильного робота TIAGo Base на основе данных ArUco-маркера была создана нода, которая отображает возможную позицию в программе RVIZ [70]. RVIZ - это инструмент для трехмерной визуализации инструментов ROS. Он обеспечивает визуализацию модели робота, информацию с датчиков робота и воспроизведение полученных данных. Он может отображать данные с камеры, лидара, с устройств 3D и 2D, включая изображения и облака точек.

При помощи различных маркеров были показаны возможные позиции робота, используя различные данные. Основные маркеры, которые отображают возможную позицию робота:

- Красный маркер – позиция робота, используя только данные с ArUco-маркера
- Синий маркер – позиция робота, используя только данные одометрии
- Желтый маркер – позиция робота, используя данные ArUco-маркера и одометрии

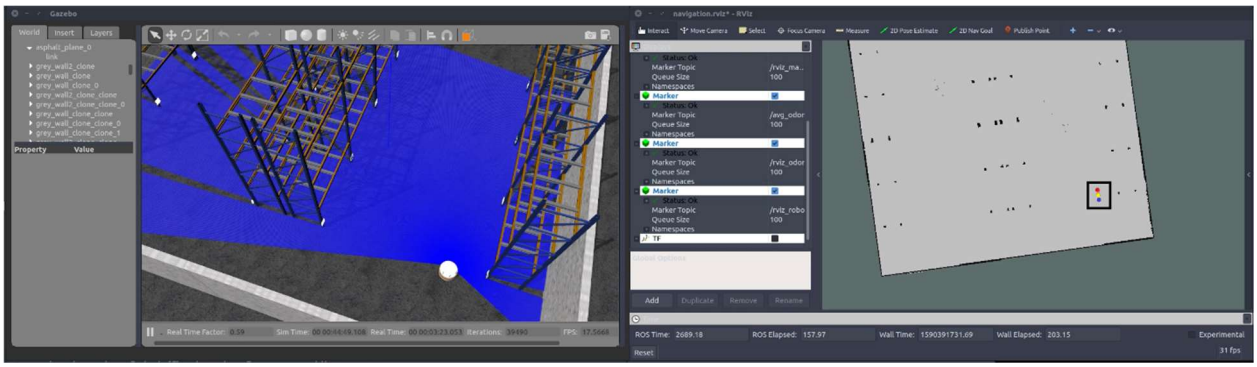


Рис. 5.4.2. Визуализация позиции робота относительно обнаруженного ArUco-маркера. Расположение робота в складском помещении в симуляторе Gazebo (слева). Возможные позиции робота относительно обнаруженного ArUco-маркера в RVIZ (справа, выделены в рамку).

5.5 ВЫРАВНИВАНИЕ МОБИЛЬНОГО РОБОТА TIAGO BASE ОТНОСИТЕЛЬНО ОБНАРУЖЕННОГО ARUCO-МАРКЕРА

Используя полученные данные трансформации ArUco-маркера, была создана программа, которая управляет скоростью мобильного робота TIAGo Base, используя proportional–integral–derivative контроллер (PID контроллер) [71]. Для управления роботом будут использоваться данные трансформации по оси X. PID контроллер имеет такие параметры, как текущее состояние, целевое значение и усилие, необходимое для достижения целевого значения. В разработанной программе целевое значение PID контроллера равно 0. Параметр состояния использует данные трансформации обнаруженного ArUco-маркера. Зная данные целевой точки и текущего состояния, PID контроллер вычисляет усилие, которое необходимо передать в топик управления скоростью мобильного робота для достижения целевой точки. Программа имеет три основные команды:

- Значение трансформации > 0, робот движется назад

- Значение трансформации < 0, робот движется вперед
- Значение трансформации = 0, робот останавливается

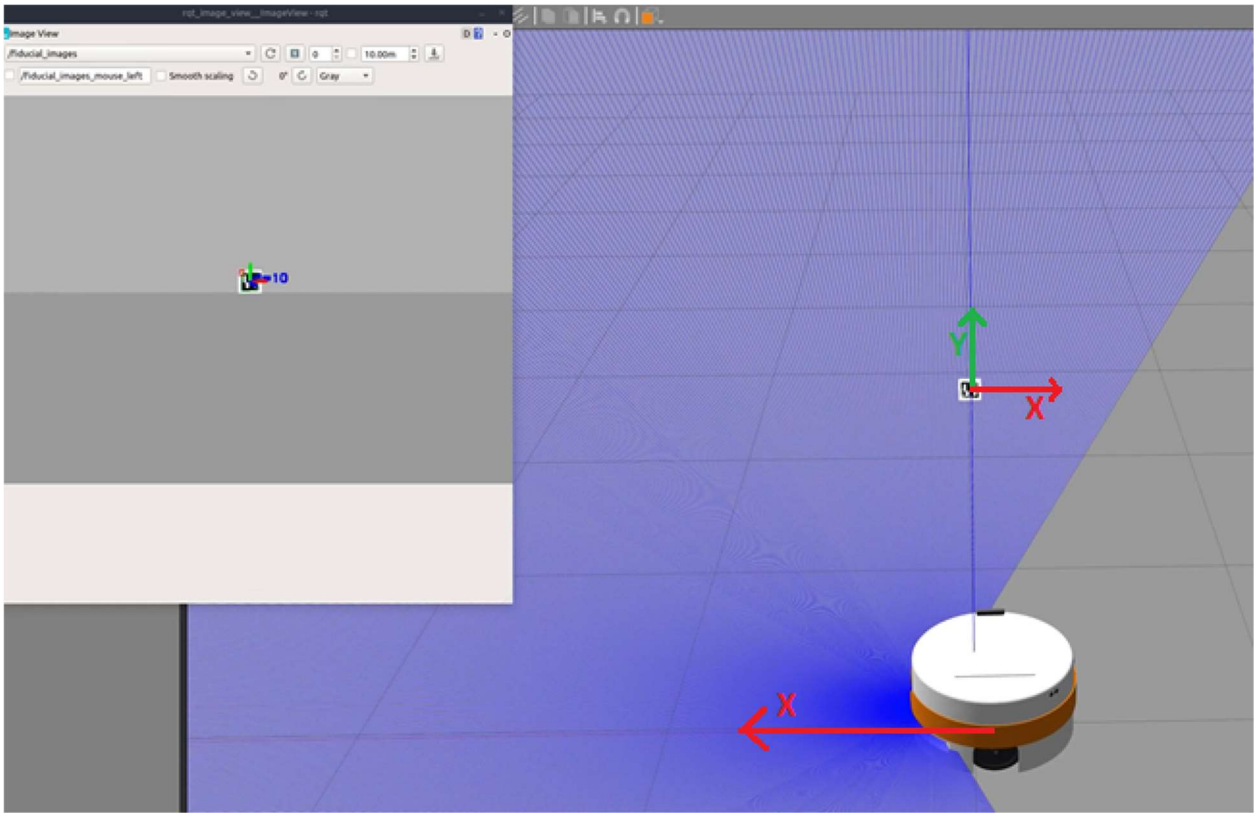


Рис. 5.5.1. Выравнивание мобильного робота TIAGo Base относительно обнаруженного ArUco-маркера.

5.6 СОЗДАНИЕ КОНТРОЛЛЕРА ДЛЯ УПРАВЛЕНИЯ РАЗРАБОТАННОЙ СИСТЕМЫ

Для управления разработанной системой автоматизации задачи инвентаризации был создан контроллер, который проверяет текущее состояние роботов и, в зависимости от полученного состояния, отправляет необходимые командные для выполнения в текущий момент времени.

Контроллер состоит из двух основных ROS-топиков:

- Action
- Feedback

Action – отправляет команды роботам для выполнения определенных действий.

Feedback – отправляет True или False, в зависимости от того выполнена поставленная задача или нет.

Контроллер работает следующим образом (рис.5.6.1):

1) Контроллер проверяет остались ли путевые точки для TIAGo Base
2) Если точки остались, то оправить путевую точку TIAGo Base
3) TIAGo Base начинает движение к путевой точке
4) Если мобильный робот обнаруживает ArUco-маркер, то вызвать ноду для выравнивания относительно обнаруженного ArUco-маркера
5) При успешном выравнивании TIAGo Base, контроллер вызывает команду для взлета БЛА PX4-LIRS
6) При достижении высоты 5 м PX4-LIRS, контроллер вызывает команду для посадки БЛА PX4-LIRS
7) Если БЛА PX4-LIRS завершил посадку успешно и выключил моторы, то контроллер проверяет список путевых точек.
8) Если в списке путевых точек остались путевые точки, то контроллер отправляет путевую точку TIAGo Base. Если в списке путевых точек не осталось, то программа завершает работу.

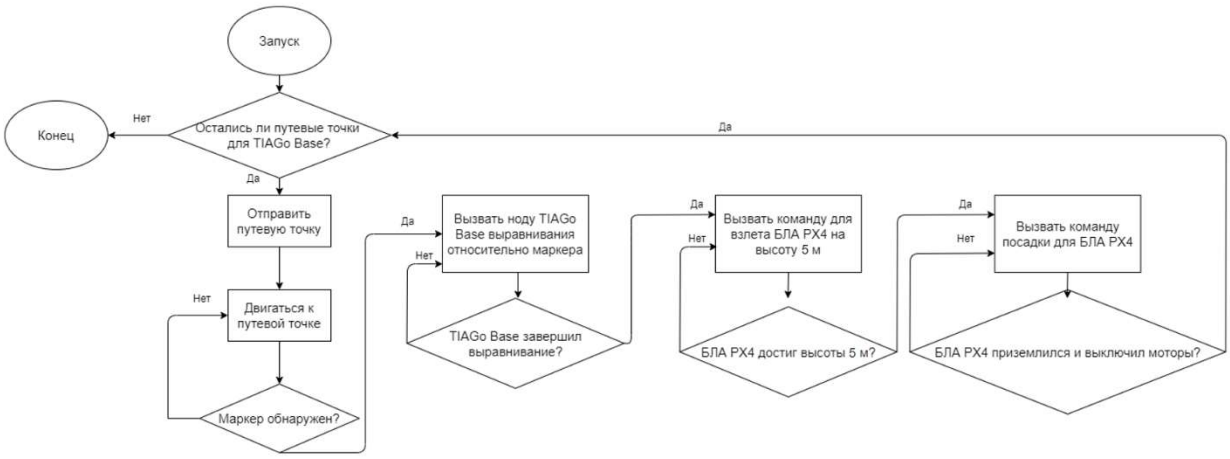


Рис. 5.6.1. Блок-схема работы алгоритма.

6. ТЕСТИРОВАНИЕ РАЗРАБОТАННОЙ СИСТЕМЫ

Для тестирования разработанной системы была использована модель складского помещения из раздела 4.3.3. Модель складского помещения была дополнительно оборудована ArUco-маркерами и RFID-метками (рис.6.1, 6.2). ArUco-маркеры необходимы для повышения точности локализации мобильного робота TIAGo Base. ArUco-маркеры были установлены на каждом стеллаже складского помещения. Общее количество ArUco-маркеров, установленных в складском помещении равно 56. Для инвентаризации складского помещения были использованы RFID-метки. Каждый товар стеллажа оборудован RFID-меткой. Общее количество установленных RFID-меток равна 504.

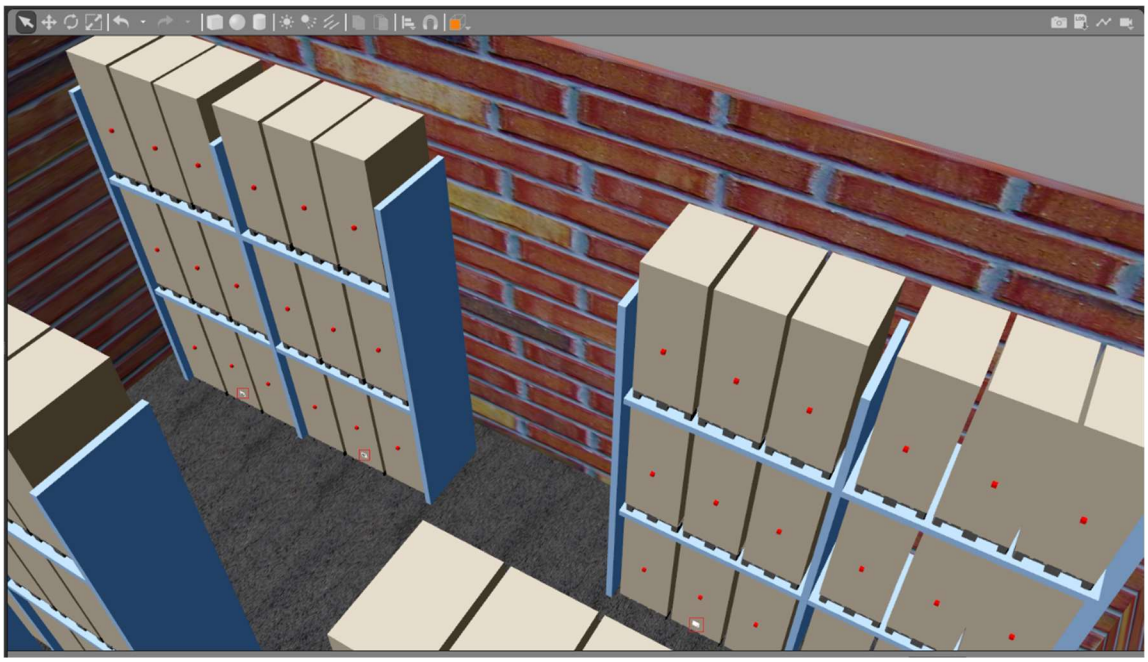


Рис.6.1. Складское помещение оборудованное ArUco-маркерами (выделено красным прямоугольником) и RFID-метками (красные кубы на каждом товаре).

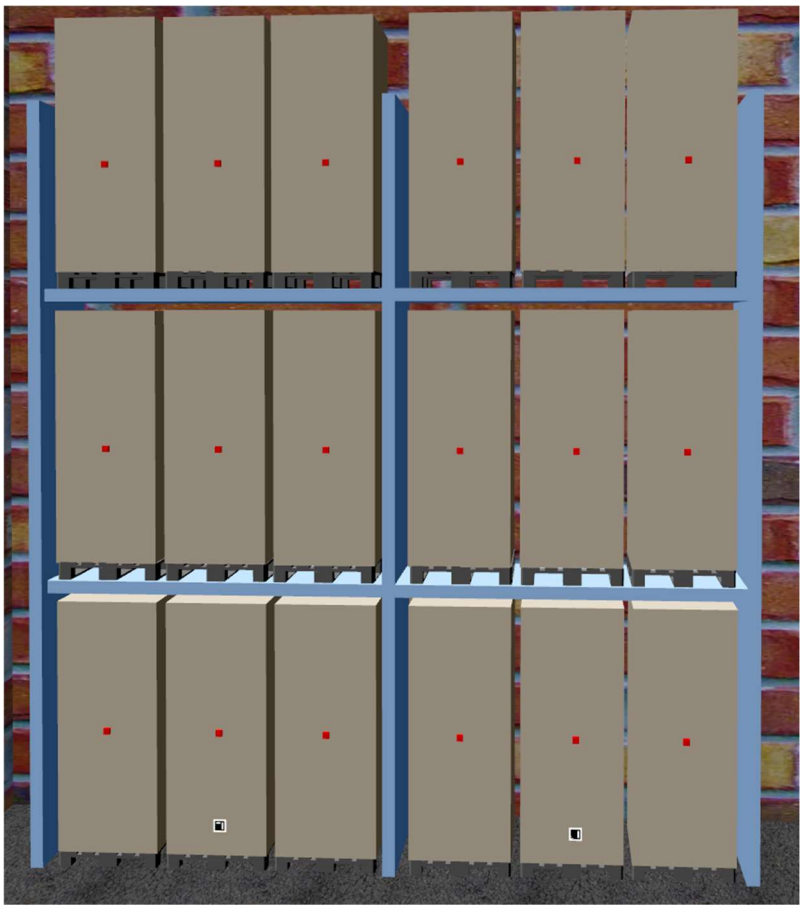


Рис.6.2. Стеллаж оборудованный ArUco-маркерами (нижний ряд, маркер расположен на центральной модели товарного ящика, над полом) и RFID-метками (красные кубы на каждом товаре).

Для автономной навигации мобильного робота TIAGo Base в созданной модели складского помещения была построена карта складского помещения (рис.6.3).

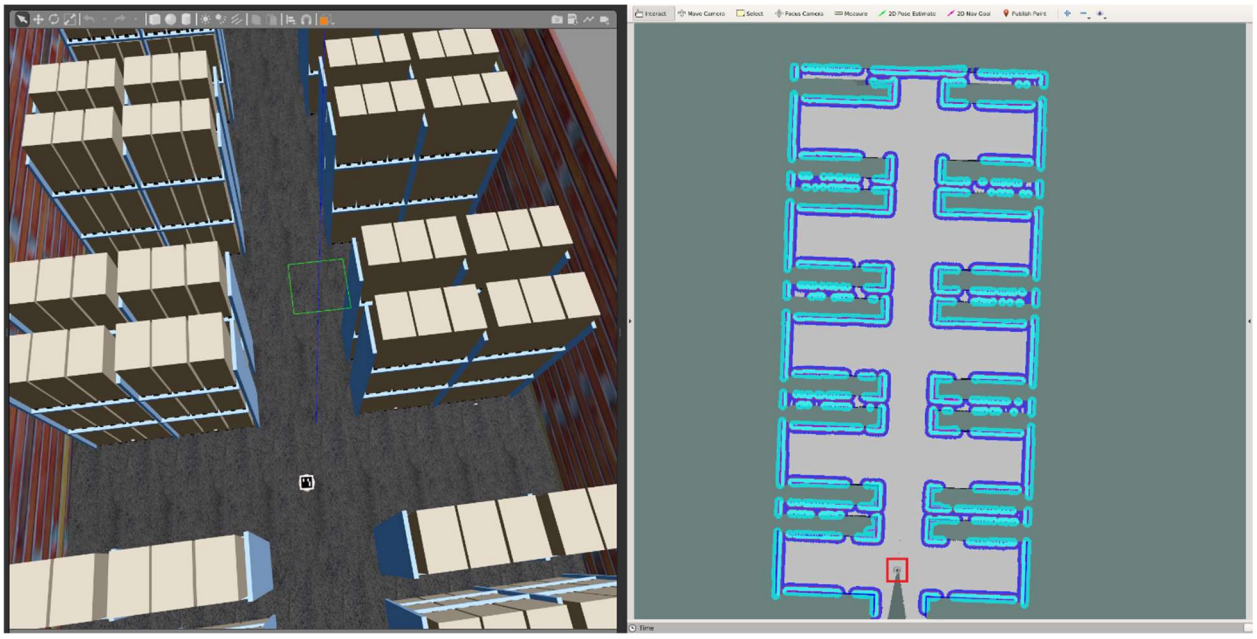


Рис. 6.3. Мобильный робот TIAGo Base в складском помещении 16.5x50.5 м в симуляторе Gazebo(слева). Карта складского помещения и позиция робота на карте в RVIZ (справа, мобильный робот TIAGo Base выделен красной рамкой).

Мобильный робот TIAGo Base оборудован ArUco-маркером для взлета и посадки БЛА PX4-LIRS (рис.6.4).

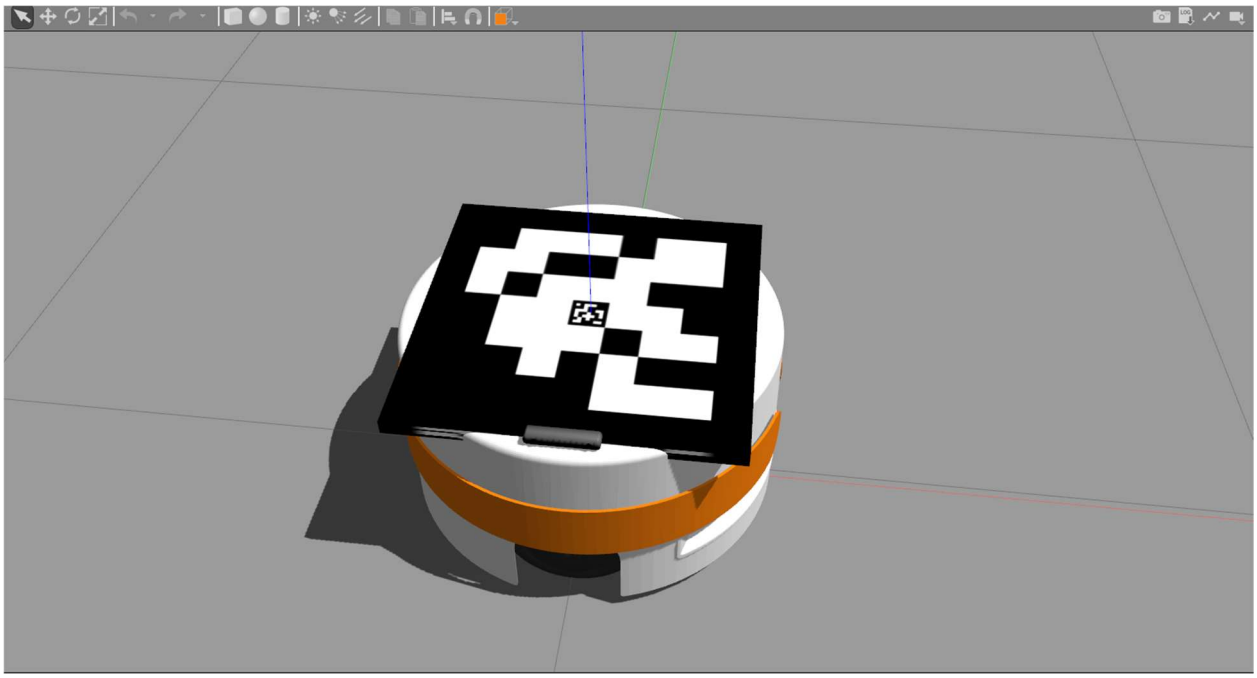


Рис. 6.4. e-ArUco маркер, установленный на поверхность мобильного робота TIAGo Base.

На БЛА PX4-LIRS был установлен RFID-считыватель для считывания обнаруженных RFID-меток в складском помещении (рис.6.5).

Рис. 6.5. БЛА PX4-LIRS оборудованный RFID-считывателем (выделен зеленым прямоугольником).

Тестирование работоспособности разработанной системы производилась в симуляторе Gazebo. Локализация мобильного робота TIAGo Base при помощи ArUco-маркеров показана на рис. 6.6. Мобильный робот TIAGo Base получает путевую точку и начинает движение к ней. При обнаружении ArUco-маркера он начинает выравниваться относительно него. После успешного выравнивания мобильного робота TIAGo Base относительно обнаруженного ArUco-маркера БЛА PX4-LIRS взлетает и позиционируется относительно ArUco-маркера установленного на TIAGo Base. Во время взлета БЛА PX4-LIRS считывает RFID-метки, которые попадают в область считывания RFID-считывателя установленного на БЛА PX4-LIRS. Далее БЛА PX4-LIRS совершает посадку на ArUco-маркер, установленный на мобильного робота TIAGo Base. При успешной посадке, мобильный робот TIAGo Base начинает движение к следующей путевой точке. Работа разработанной системы завершится, когда будут достигнуты все путевые точки.

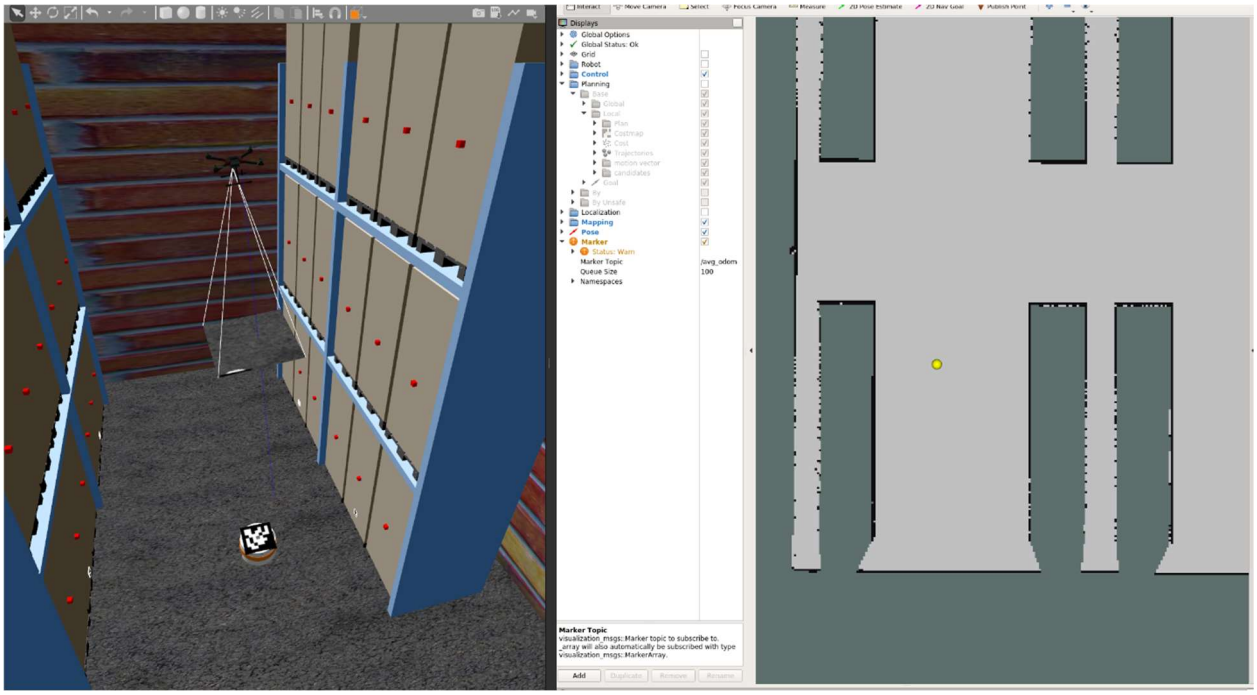


Рис. 6.6. Взлет и инвентаризация RFID-меток в симуляторе Gazebo(слева). Локализация мобильного робота TIAGo Base относительно обнаруженного ArUco-маркера в RVIZ (справа).

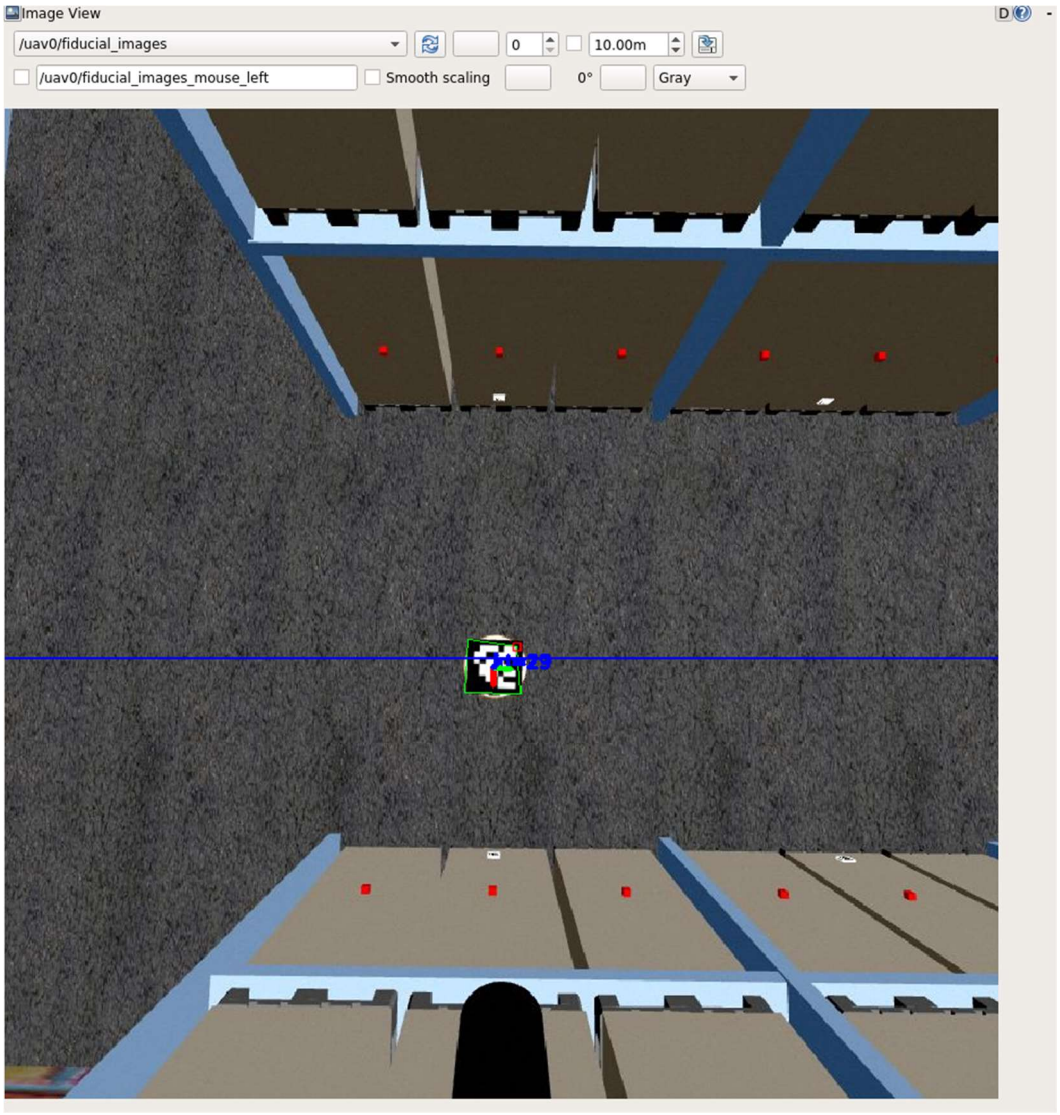


Рис. 6.7. Обнаружение e-ArUco маркера установленного на поверхность мобильного робота TIAGo Base.

Рис. 6.8. Обнаруженные RFID-метки.

7. ЭКСПЕРИМЕНТЫ

Для проверки ноды выравнивания TIAGo Base относительно обнаруженного ArUco-маркера был использован реальный робот. На мобильный робот TIAGo Base была установлена камера для обнаружения ArUco-маркеров. При обнаружении ArUco-маркера TIAGo Base выравнивается относительно его центра, если ArUco-маркер не обнаружен, то TIAGo Base ожидает пока ArUco-маркер не будет в зоне его обнаружения.

Рис. 7.1. Проверка работы ноды выравнивания мобильного робота TIAGo Base относительно обнаруженного ArUco-маркера.

ЗАКЛЮЧЕНИЕ

В данной работе представлена новая система автоматизации задач инвентаризации складских помещений с использование локализации наземного робота-доставщика при помощи ArUco-маркеров. Инвентаризацию складского помещения производит БЛА PX4-LIRS, используя установленный на него RFID-считыватель. Роль наземного робота-доставщика исполняет мобильный робот TIAGo Base. Для автономной навигации и локализации мобильный робот TIAGo Base строит карту складского помещения, при помощи установленных на него сенсоров. Мобильный робот TIAGo Base использует данные трансформации обнаруженных ArUco-маркеров, установленных на каждом стеллаже складского помещения для повышения точности локализации в складском помещении.

Дополнительно были созданы модели складских помещений по стандартам ГОСТ для использования в среде ROS/Gazebo. Каждый элемент складского помещения: размеры и тип стеллажа, расстановка товаров на стеллаже и расстановка стеллажей в складском помещении создана в соответствии с регламентами ГОСТ, которые описывают все необходимые параметры для каждого элемента складского помещения. Созданные модели повысят качество тестирования созданной системы перед развертыванием на реальных роботах.

Также был предложен новый тип ArUco-маркеров, который был назван e-ArUco. Он имеет дополнительный внутренний маркер в центре исходного (внешнего) ArUco-маркера. Два маркера (внешний и внутренний) имеют разные идентификаторы. Особенностью данного типа маркеров позволит повысить точность приземления БЛА PX4-LIRS на наземного робота-доставщика, т.к. e-ArUco маркер может быть обнаружен на протяжении всего процесса посадки БЛА PX4-LIRS.

В дальнейшем планируется разработать базу данных, в котором будут храниться, обновляться и обрабатываться информация обнаруженных меток по установленным правилам. Также планируется добавить карту расположения обнаруженных RFID-меток.

На основании проведенных исследований и разработок было опубликовано 3 статьи [43;72;73], сопровождаемые докладами на международных конференциях, таких как Climbing and Walking Robots and Support Technologies for Mobile Machines (CLAWAR 2020), Conference on Developments in eSystems Engineering (DeSE 2020) и Siberian Conference on Control and Communications (SIBCON 2021).

Исходный код и файл данной ВКР опубликован в репозитории ЛИРС на платформе GitLab: https://gitlab.com/LIRS_Projects/simulation-fiducial-slam.

СПИСОК ИСПОЛЬЗУЕМЫХ ИСТОЧНИКОВ

1. Bogue R. Growth in e-commerce boosts innovation in the warehouse robot market //Industrial Robot: An International Journal. – 2016.
2. TIAGo Base [Электронный ресурс] / TIAGo Base overview —Режим доступа — URL: https://pal-robotics.com/robots/tiago-base/ (дата обращения: 28.05.2021)
3. PAL-Robotics [Электронный ресурс] / PAL-Robotics official page — Режим доступа — URL: http://pal-robotics.com/ (дата обращения: 28.05.2021)
4. Pixhawk [Электронный ресурс] / Pixhawk — Режим доступа — URL: http://pixhawk.org/ (дата обращения: 28.05.2021)
5. Finkenzeller K. RFID handbook: fundamentals and applications in contactless smart cards, radio frequency identification and near-field communication. – John Wiley & Sons, 2010. – С. 21-74.
6. Robot Operating System (ROS) [Электронный ресурс] / ROS Melodic documentation — Режим доступа — URL: http://wiki.ros.org/ (дата обращения: 28.05.2021)
7. Gazebo simulation [Электронный ресурс] / Gazebo 9.0 documentation —Режим доступа — URL: http://gazebosim.org/ (дата обращения: 28.05.2021)
8. Beul M. et al. Fast autonomous flight in warehouses for inventory applications //IEEE Robotics and Automation Letters. – 2018. – Т. 3. – №. 4. – С. 3121-3128.
9. Cho H. et al. 2D barcode detection using images for drone-assisted inventory management //2018 15th International Conference on Ubiquitous Robots (UR). – IEEE, 2018. – С. 461-465.

10. Ehrenberg I., Floerkemeier C., Sarma S. Inventory management with an RFID-equipped mobile robot //2007 IEEE International Conference on Automation Science and Engineering. – IEEE, 2007. – С. 1020-1026.
11. Zhang J. et al. Mobile robot for retail inventory using RFID //2016 IEEE international conference on Industrial technology (ICIT). – IEEE, 2016. – С. 101-106.
12. Bae S. M. et al. Development of inventory checking system based on UAV and RFID in open storage yard //2016 International Conference on Information Science and Security (ICISS). – IEEE, 2016. – С. 1-2.
13. Ong J. H., Sanchez A., Williams J. Multi-uav system for inventory automation //2007 1st Annual RFID Eurasia. – IEEE, 2007. – С. 1-6.
14. Kwon W. et al. Robust autonomous navigation of unmanned aerial vehicles (UAVs) for warehouses' inventory application //IEEE Robotics and Automation Letters. – 2019. – Т. 5. – №. 1. – С. 243-249.
15. De Falco A., Narducci F., Petrosino A. An UAV autonomous warehouse inventorying by deep learning //International Conference on Image Analysis and Processing. – Springer, Cham, 2019. – С. 443-453.
16. Cristiani D. et al. Inventory Management through Mini-Drones: Architecture and Proof-of-Concept Implementation //2020 IEEE 21st International Symposium on" A World of Wireless, Mobile and Multimedia Networks"(WoWMoM). – IEEE, 2020. – С. 317-322.
17. Guérin F. et al. Towards an autonomous warehouse inventory scheme //2016 IEEE Symposium Series on Computational Intelligence (SSCI). – IEEE, 2016. – С. 1-8.
18. Beul M. et al. Autonomous navigation in a warehouse with a cognitive micro aerial vehicle //Robot Operating System (ROS). – Springer, Cham, 2017. – С. 487-524.
19. Kalinov I. et al. Warevision: Cnn barcode detection-based uav trajectory optimization for autonomous warehouse stocktaking //IEEE Robotics and Automation Letters. – 2020. – Т. 5. – №. 4. – С. 6647-6653.

20. Alam F., Hasan K. S. B., Varshney A. Low-Cost Autonomous Vehicle for Inventory Movement in Warehouses //2020 Sixth International Conference on Parallel, Distributed and Grid Computing (PDGC). – IEEE, 2020. – С. 400-405.
21. Складская робототехника [Электронный ресурс] / What is warehouse robotics? — Режим доступа — URL: https://6river.com/what-is-warehouse-robotics/ (дата обращения: 28.05.2021)
22. Babinec A. et al. Visual localization of mobile robot using artificial markers //Procedia Engineering. – 2014. – Т. 96. – С. 1-9.
23. Willow Garage [Электронный ресурс] / Willow Garage — Режим доступа — URL: http://www.willowgarage.com/ (дата обращения: 28.05.2021)
24. OGRE [Электронный ресурс] / OGRE — Режим доступа — URL: https://www.ogre3d.org/ (дата обращения: 28.05.2021)
25. ODE [Электронный ресурс] / Open Dynamics Engine — Режим доступа — URL: https://www.ode.org/ (дата обращения: 28.05.2021)
26. Simbody [Электронный ресурс] / Simbody: Multibody Physics API — Режим доступа — URL: https://simtk.org/projects/simbody/ (дата обращения: 28.05.2021)
27. Dynamic Animation and Robotics Toolkit [Электронный ресурс] / DART — Режим доступа — URL: https://dartsim.github.io/ (дата обращения: 28.05.2021)
28. ros_control [Электронный ресурс] / ros_control documentation — Режим доступа — URL: http://wiki.ros.org/ros_control/ (дата обращения: 28.05.2021)
29. TIAGo Base [Электронный ресурс] / ROS Melodic, TIAGo Base package — Режим доступа — URL: http://wiki.ros.org/Robots/PMB-2 (дата обращения: 28.05.2021)
30. PX4 Development Guide [Электронный ресурс] / PX4 — Режим доступа — URL: https://dev.px4.io/en/ (дата обращения: 28.05.2021)

31. ArduPilot [Электронный ресурс] / ArduPilot documentation — Режим доступа — URL: https://ardupilot.org/ (дата обращения: 28.05.2021)
32. QGroundControl [Электронный ресурс] / QGroundControl documentation — Режим доступа — URL: http://qgroundcontrol.com/ (дата обращения: 28.05.2021)
33. Koubaa A. et al. Micro air vehicle link (mavlink) in a nutshell: A survey //IEEE Access. – 2019. – Т. 7. – С. 87658-87680.
34. Dronecode [Электронный ресурс] / Dronecode — Режим доступа — URL: https://www.dronecode.org/ (дата обращения: 28.05.2021)
35. MAVROS [Электронный ресурс] / ROS Melodic, MAVROS package — Режим доступа — URL: http://wiki.ros.org/mavros/ (дата обращения: 28.05.2021)
36. Классификация складских помещений [Электронный ресурс] / Knight Frank — Режим доступа — URL: https://content.knightfrank.com/resources/knightfrank.ru/pdf/research/ind.pdf (дата обращения: 28.05.2021)
37. ГОСТ Р 55525-2013 Складское оборудование. Стеллажи сборно-разборные. Общие технические условия [Электронный ресурс] / Электронный Фонд — Режим доступа — URL: http://docs.cntd.ru/document/1200103726 (дата обращения: 28.05.2021)
38. ГОСТ 16140-77 Стеллажи сборно-разборные. Технические условия [Электронный ресурс] / Gosthelp — Режим доступа — URL: http://www.gosthelp.ru/text/GOST1614077Stellazhisborn.html (дата обращения: 28.05.2021)
39. AWS Robotics [Электронный ресурс] / AWS Robotics Warehouse — Режим доступа — URL: https://github.com/aws-robotics/aws-robomaker-small-warehouse-world/ (дата обращения: 28.05.2021)
40. Greco G. et al. Localization of RFID tags for environmental monitoring using UAV //2015 IEEE 1st International Forum on Research and

Technologies for Society and Industry Leveraging a better tomorrow (RTSI). – IEEE, 2015. – С. 480-483.

41. Abdelrasoul Y., Saman A. B. S. H. M., Sebastian P. A quantitative study of tuning ROS gmapping parameters and their effect on performing indoor 2D SLAM //2016 2nd IEEE International Symposium on Robotics and Manufacturing Automation (ROMA). – IEEE, 2016. – С. 1-6.
42. Thingmagic_usbpro [Электронный ресурс] / ROS Kinetic, thingmagic_usbpro package — Режим доступа — URL: http://wiki.ros.org/thingmagic_usbpro (дата обращения: 28.05.2021)
43. Khazetdinov A. et al. RFID-based warehouse management system prototyping using a heterogeneous team of robots //Robots in Human Life. – с. 263.
44. PX4Flow Smart Camera [Электронный ресурс] / px4flow documentation — Режим доступа — URL: https://docs.px4.io/master/en/sensor/px4flow.html (дата обращения: 28.05.2021)
45. MB1242 [Электронный ресурс] / MB1242 documentation — Режим доступа — URL: http://maxbotix.com/ultrasonic_sensors/mb1242.htm (дата обращения: 28.05.2021)
46. Zakiev A. et al. Pilot virtual experiments on aruco and artag systems comparison for fiducial marker rotation resistance //Proceedings of 14th International Conference on Electromechanics and Robotics “Zavalishin's Readings”. – Springer, Singapore, 2020. – С. 455-464.
47. Liu D., Yu J. Otsu method and K-means //2009 Ninth International Conference on Hybrid Intelligent Systems. – IEEE, 2009. – Т. 1. – С. 344-349.
48. Kingston D., Rasmussen S., Humphrey L. Automated UAV tasks for search and surveillance //2016 IEEE Conference on Control Applications (CCA). – IEEE, 2016. – С. 1-8.

49. Danilov A. S., Smirnov U. D., Pashkevich M. A. The system of the ecological monitoring of environment which is based on the usage of UAV //Russian journal of ecology. – 2015. – Т. 46. – №. 1. – С. 14-19.
50. Sankarasrinivasan S. et al. Health monitoring of civil structures with integrated UAV and image processing system //Procedia Computer Science. – 2015. – Т. 54. – С. 508-515.
51. Shakhatreh H. et al. Unmanned aerial vehicles (UAVs): A survey on civil applications and key research challenges //Ieee Access. – 2019. – Т. 7. – С. 48572-48634.
52. Levonevskiy D., Vatamaniuk I., Saveliev A. Providing Availability of the Smart Space Services by Means of Incoming Data Control Methods //International Conference on Interactive Collaborative Robotics. – Springer, Cham, 2018. – С. 170-180.
53. Emel'yanov S. et al. Multilayer cognitive architecture for UAV control //Cognitive Systems Research. – 2016. – Т. 39. – С. 58-72.
54. Babinec A. et al. Visual localization of mobile robot using artificial markers //Procedia Engineering. – 2014. – Т. 96. – С. 1-9.
55. Irfan M. et al. Vision-based Guidance and Navigation for Autonomous MAV in Indoor Environment //2020 11th International Conference on Computing, Communication and Networking Technologies (ICCCNT). – IEEE, 2020. – С. 1-5.
56. Mingachev E. et al. Comparison of ROS-Based Monocular Visual SLAM Methods: DSO, LDSO, ORB-SLAM2 and DynaSLAM //International Conference on Interactive Collaborative Robotics. – Springer, Cham, 2020. – С. 222-233.
57. Tørdal S. S., Hovland G. Relative vessel motion tracking using sensor fusion, aruco markers, and mru sensors. – 2017.
58. Garrido-Jurado S. et al. Automatic generation and detection of highly reliable fiducial markers under occlusion //Pattern Recognition. – 2014. – Т. 47. – №. 6. – С. 2280-2292.

59. Oščádal P. et al. Improved Pose Estimation of Aruco Tags Using a Novel 3D Placement Strategy //Sensors. – 2020. – Т. 20. – №. 17. – С. 4825.
60. Sagitov A. et al. Comparing fiducial marker systems in the presence of occlusion //2017 International Conference on Mechanical, System and Control Engineering (ICMSC). – IEEE, 2017. – С. 377-382.
61. aruco_detect [Электронный ресурс] / aruco_detect documentation — Режим доступа — URL: http://wiki.ros.org/aruco_detect/ (дата обращения: 28.05.2021)
62. Lebedev I., Erashov A., Shabanova A. Accurate Autonomous UAV Landing Using Vision-Based Detection of ArUco-Marker //International Conference on Interactive Collaborative Robotics. – Springer, Cham, 2020. – С. 179-188.
63. Wubben J. et al. Accurate landing of unmanned aerial vehicles using ground pattern recognition //Electronics. – 2019. – Т. 8. – №. 12. – С. 1532.
64. Wynn J. S., McLain T. W. Visual servoing for multirotor precision landing in daylight and after-dark conditions //2019 International Conference on Unmanned Aircraft Systems (ICUAS). – IEEE, 2019. – С. 1242-1248.
65. Santos H. B. et al. Control of mobile robots using actionlib //Robot Operating System (ROS). – Springer, Cham, 2017. – С. 161-189.
66. move_base [Электронный ресурс] / move_base documentation — Режим доступа — URL: http://wiki.ros.org/move_base/ (дата обращения: 28.05.2021)
67. depthimage_to_laserscan [Электронный ресурс] / depthimage_to_laserscan documentation — Режим доступа — URL: http://wiki.ros.org/depthimage_to_laserscan/ (дата обращения: 28.05.2021)

68. Peng G. et al. An improved AMCL algorithm based on laser scanning match in a complex and unstructured environment //Complexity. – 2018. – T. 2018.
69. Welch G. et al. An introduction to the Kalman filter. – 1995.
70. Kam H. R. et al. Rviz: a toolkit for real domain data visualization //Telecommunication Systems. – 2015. – T. 60. – №. 2. – C. 337-345.
71. Ang K. H., Chong G., Li Y. PID control system analysis, design, and technology //IEEE transactions on control systems technology. – 2005. – T. 13. – №. 4. – C. 559-576.
72. Khazetdinov A. et al. Standart-complaint Gazebo warehouse modelling and validation // 2020 13th International Conference on Developments in eSystems Engineering (DeSE). – IEEE, 2021. – C. 1-4.
73. Khazetdinov A. et al. Embedded ArUco: a novel approach for high precision UAV landing //2021 International Siberian Conference on Control and Communications (SIBCON). – IEEE, 2021. – C. 1-6.

ПРИЛОЖЕНИЕ А

Взлет и посадка на e-ArUco маркер

```
#include <fiducial_msgs/FiducialTransform.h>
#include <fiducial_msgs/FiducialTransformArray.h>
#include <geometry_msgs/PoseStamped.h>
#include <geometry_msgs/TwistStamped.h>
#include <mavros_msgs/CommandBool.h>
#include <mavros_msgs/SetMode.h>
#include <mavros_msgs/State.h>
#include <std_msgs/Float64.h>
#include "ros/ros.h"
#include "std_msgs/Bool.h"
#include "std_msgs/builtin_int8.h"
#include "sensor_msgs/Range.h"

mavros_msgs::State current_state;
geometry_msgs::PoseStamped pose, current_pose, land_pose;
std_msgs::Int8 command;
fiducial_msgs::FiducialTransformArray marker_pose;
geometry_msgs::TwistStamped cmd, current_cmd;
```

```
std_msgs::Float64 setpoint, state_x, state_y, state_z, setpoint_z;
mavros_msgs::CommandBool arm_cmd;
std_msgs::Bool px4_takeoff, px4_land, px4_feedback;
bool t = true;
bool t1 = true;
double pre_err_x = 0;
double pre_err_y = 0;
double control_effort_x = 0.0;
double control_effort_y = 0.0;
double control_effort_z = 0.0;
double fake_h = 0;
sensor_msgs::Range mb1242;

void follow_marker();
void takeoff();
void land();
void wait();
void follow_marker_land();

void state_cb(const mavros_msgs::State::ConstPtr &msg)
{
 current_state = *msg;
}

void handle_pose(const geometry_msgs::PoseStamped &_pose)
{
 current_pose = _pose;
}
void handle_cmd(const geometry_msgs::TwistStamped &_cmd)
{
 current_cmd = _cmd;
}

void handle_command(const std_msgs::Int8 &_command)
{
 command = _command;
}
void pose_cb(const fiducial_msgs::FiducialTransformArrayConstPtr &msg)
{
 marker_pose = *msg;
}
void controlEffortCallbackX(const std_msgs::Float64 &control_effort_input)
{
 control_effort_x = control_effort_input.data;
}
void controlEffortCallbackY(const std_msgs::Float64 &control_effort_input)
{
 control_effort_y = control_effort_input.data;
}
void controlEffortCallbackZ(const std_msgs::Float64 &control_effort_input)
{
 control_effort_z = control_effort_input.data;
}
void mb1242_cb(const sensor_msgs::RangeConstPtr &msg)
{
 mb1242 = *msg;
}

int main(int argc, char **argv)
{
 ros::init(argc, argv, "offb_node");
 ros::NodeHandle nh;
```

```
  ros::Subscriber state_sub = nh.subscribe<mavros_msgs::State>("uav0/mavros/state", 10, state_cb);
  ros::Publisher local_pos_pub =
nh.advertise<geometry_msgs::PoseStamped>("uav0/mavros/setpoint_position/local", 10);
  ros::ServiceClient arming_client = nh.serviceClient<mavros_msgs::CommandBool>("uav0/mavros/cmd/arming");
  ros::ServiceClient set_mode_client = nh.serviceClient<mavros_msgs::SetMode>("uav0/mavros/set_mode");
  ros::Subscriber pose_sub = nh.subscribe("uav0/mavros/local_position/pose", 1000, handle_pose);
  ros::Subscriber command_sub = nh.subscribe("/pixhawk_quad/command", 1000, handle_command);
  ros::Subscriber pose_sb = nh.subscribe<fiducial_msgs::FiducialTransformArray>("uav0/fiducial_transforms", 10,
pose_cb);
  ros::Publisher takeoff_pub = nh.advertise<std_msgs::Bool>("robots_feedback/px4_takeoff", 10);
  ros::Publisher land_pub = nh.advertise<std_msgs::Bool>("robots_feedback/px4_land", 10);
  ros::Publisher px4_feedback_pub = nh.advertise<std_msgs::Bool>("robots_feedback/px4_feedback", 10);
  // PID
  ros::Publisher setpoint_pub = nh.advertise<std_msgs::Float64>("setpoint", 1);
  ros::Publisher state_x_pub = nh.advertise<std_msgs::Float64>("uav0_x/state", 1);
  ros::Subscriber sub_x = nh.subscribe("uav0_x/control_effort", 1, controlEffortCallbackX);
  ros::Publisher state_y_pub = nh.advertise<std_msgs::Float64>("uav0_y/state", 1);
  ros::Subscriber sub_y = nh.subscribe("uav0_y/control_effort", 1, controlEffortCallbackY);
  ros::Publisher state_z_pub = nh.advertise<std_msgs::Float64>("uav0_z/state", 1);
  ros::Subscriber sub_z = nh.subscribe("uav0_z/control_effort", 1, controlEffortCallbackZ);
  ros::Publisher setpoint_z_pub = nh.advertise<std_msgs::Float64>("z/setpoint", 1);

  ros::Subscriber mb1242_sub = nh.subscribe<sensor_msgs::Range>("mavros/px4flow/ground_distance", 1,
mb1242_cb);

  ros::Rate rate(20.0);
  command.data = 0;
  setpoint.data = 0;
  setpoint_z.data = 2;
  bool is_takeoff = false;
  px4_takeoff.data = false;
  px4_land.data = false;
  px4_feedback.data = false;

  while (ros::ok() && !current_state.connected)
  {
   ros::spinOnce();
   rate.sleep();
  }
  for (int i = 100; ros::ok() && i > 0; --i)
  {
   local_pos_pub.publish(current_pose);
   ros::spinOnce();
   rate.sleep();
  }
  while (ros::ok())
  {
   switch (command.data)
   {
   case 0:
     break;
   case 1:
     takeoff();
     is_takeoff = true;
     break;
   case 2:
     if (is_takeoff)
       follow_marker();
     break;
   case 3:
     land();
     is_takeoff = false;
```

```
      break;
    case 4:
      wait();
      break;
    default:
      break;
    }
    setpoint_z_pub.publish(setpoint_z);
    setpoint_pub.publish(setpoint);
    state_x_pub.publish(state_x);
    state_y_pub.publish(state_y);
    state_z_pub.publish(state_z);
    local_pos_pub.publish(pose);
    takeoff_pub.publish(px4_takeoff);
    land_pub.publish(px4_land);
    px4_feedback_pub.publish(px4_feedback);

    ros::spinOnce();
    rate.sleep();
  }
  return 0;
}

void takeoff()
{
  ros::NodeHandle nh;
  ros::ServiceClient set_mode_client = nh.serviceClient<mavros_msgs::SetMode>("uav0/mavros/set_mode");
  ros::ServiceClient arming_client = nh.serviceClient<mavros_msgs::CommandBool>("uav0/mavros/cmd/arming");
  ros::Rate rate(20.0);
  mavros_msgs::SetMode offb_set_mode;
  ros::Time last_request = ros::Time::now();
  offb_set_mode.request.custom_mode = "OFFBOARD";
  bool is_takeoff = false;
  px4_feedback.data = false;

  if (current_state.mode != "OFFBOARD")
  {
    if (set_mode_client.call(offb_set_mode) && offb_set_mode.response.mode_sent)
    {
      // ROS_INFO("Offboard enabled");
    }
    last_request = ros::Time::now();
  }
  else
  {
    if (!current_state.armed)
    {
      if (arming_client.call(arm_cmd) && arm_cmd.response.success)
      {
        // ROS_INFO("Vehicle armed");
      }
      last_request = ros::Time::now();
    }
  }
  arm_cmd.request.value = true;

  for (int i = 0; i < marker_pose.transforms.size(); i++)
  {
    state_x.data = marker_pose.transforms[i].transform.translation.x;
    state_y.data = marker_pose.transforms[i].transform.translation.y;
    state_z.data = mb1242.range;
```

```
    t = true;
    t1 = true;
    ROS_INFO("CONTROL X = %f", control_effort_x);
    ROS_INFO("CONTROL Y = %f", control_effort_y);
    ROS_INFO("CONTROL Z = %f", control_effort_z);

    // TAKEOFF FROM ARUCO

    if (abs(marker_pose.transforms[i].transform.translation.x) <= 0.05 &&
        abs(marker_pose.transforms[i].transform.translation.y) <= 0.05)
    {
      pose.pose.position.z = mb1242.range + control_effort_z;
      fake_h = mb1242.range + control_effort_z;
    }
    else
    {
      follow_marker();
      pose.pose.position.z = fake_h;
    }
    if (marker_pose.transforms[i].transform.translation.z > 1.7)
    {
      px4_takeoff.data = true;
    }
  }
}

void land()
{
  ros::NodeHandle nh;
  ros::ServiceClient set_mode_client = nh.serviceClient<mavros_msgs::SetMode>("/uav0/mavros/set_mode");
  ros::ServiceClient arming_client = nh.serviceClient<mavros_msgs::CommandBool>("/uav0/mavros/cmd/arming");
  mavros_msgs::SetMode offb_set_mode;
  mavros_msgs::CommandBool arm_cmd;
  arm_cmd.request.value = false;

  for (int i = 0; i < marker_pose.transforms.size(); i++)
  {
    // state_z.data = marker_pose.transforms[i].transform.translation.z;

    if (marker_pose.transforms[i].fiducial_id == 29 && abs(marker_pose.transforms[i].transform.translation.x) <=
0.05 &&
        abs(marker_pose.transforms[i].transform.translation.y) <= 0.05)
    {
      pose.pose.position.z -= 0.01;
    }
    else
    {
      follow_marker_land();
    }
    if (marker_pose.transforms[i].fiducial_id == 33 && abs(marker_pose.transforms[i].transform.translation.x) <=
0.02 &&
        abs(marker_pose.transforms[i].transform.translation.y) <= 0.02)
    {
      if (abs(marker_pose.transforms[i].transform.translation.z) >= 0.4)
      {
        pose.pose.position.z -= 0.01;
      }
      else
      {
        land_pose.pose.position.x = current_pose.pose.position.x;
        land_pose.pose.position.y = current_pose.pose.position.y;
        t1 = false;
```

```
      }
    }
    else if (t1)
    {
      follow_marker_land();
    }
  }
  if (!t1)
  {
    if (current_state.mode != "AUTO.LAND")
    {
      offb_set_mode.request.custom_mode = "AUTO.LAND";
      arming_client.call(arm_cmd);
    }
    if (set_mode_client.call(offb_set_mode) && offb_set_mode.response.mode_sent && current_state.mode ==
"AUTO.LAND")
    {
      ROS_INFO("DISARMING!");
      arming_client.call(arm_cmd);
      px4_land.data = true;
    }
  }
}

void wait()
{
  px4_land.data = false;
  px4_takeoff.data = false;
  px4_feedback.data = true;
}
void follow_marker()

{
  for (int i = 0; i < marker_pose.transforms.size(); i++)
  {
    state_x.data = marker_pose.transforms[i].transform.translation.x;

    state_y.data = marker_pose.transforms[i].transform.translation.y;
    state_z.data = mb1242.range;
    if (state_y.data < 0)
    {
      pose.pose.position.x = current_pose.pose.position.x + abs(control_effort_y);
    }
    else
    {
      pose.pose.position.x = current_pose.pose.position.x - abs(control_effort_y);
    }
    if (state_x.data < 0)
    {
      pose.pose.position.y = current_pose.pose.position.y + abs(control_effort_x);
    }
    else
    {
      pose.pose.position.y = current_pose.pose.position.y - abs(control_effort_x);
    }
    pose.pose.position.z = mb1242.range + control_effort_z;
  }
}
void follow_marker_land()

{
  for (int i = 0; i < marker_pose.transforms.size(); i++)
```

```
  {
   state_x.data = marker_pose.transforms[i].transform.translation.x;

   state_y.data = marker_pose.transforms[i].transform.translation.y;
   state_z.data = mb1242.range;
   if (state_y.data < 0)
   {
    pose.pose.position.x = current_pose.pose.position.x + abs(control_effort_y);
   }
   else
   {
    pose.pose.position.x = current_pose.pose.position.x - abs(control_effort_y);
   }
   if (state_x.data < 0)
   {
    pose.pose.position.y = current_pose.pose.position.y + abs(control_effort_x);
   }
   else
   {
    pose.pose.position.y = current_pose.pose.position.y - abs(control_effort_x);
   }
  }
}
```

ПРИЛОЖЕНИЕ Б

Выравнивание мобильного робота TIAGo Base относительно ArUco-маркера

```
#include <fiducial_msgs/FiducialTransformArray.h>
#include <geometry_msgs/Twist.h>
#include <nav_msgs/Odometry.h>
#include <ros/ros.h>
#include <std_msgs/Bool.h>
#include <std_msgs/Float64.h>
fiducial_msgs::FiducialTransformArray aruco_pose;
geometry_msgs::Twist cmd;
std_msgs::Float64 state, setpoint;
std_msgs::Bool pmb2_follow;
int k = 0;
float c = 1;
float c1 = 1;
int last_id = 999;
bool rot_flag = true;
double control_effort = 0;
bool feedback = false;
void pose_aruco_cb(const fiducial_msgs::FiducialTransformArrayConstPtr& msg)
{
 aruco_pose = *msg;
}
void controlEffortCallback(const std_msgs::Float64& control_effort_input)
{
 control_effort = control_effort_input.data;
}
void pmb2_follow_cb(const std_msgs::BoolConstPtr& msg)
{
 pmb2_follow = *msg;
 // ROS_INFO("FRS %s", pmb2_follow.data ? "true" : "false");
}
```

```
int main(int argc, char** argv)
{
 ros::init(argc, argv, "aruco_follow");
 ros::NodeHandle n;
 ros::Rate r(1);
 bool action = false;
 ros::Subscriber pose_aruco_sb =
    n.subscribe<fiducial_msgs::FiducialTransformArray>("/fiducial_transforms", 1000, pose_aruco_cb);
 ros::Publisher cmd_vel_pub = n.advertise<geometry_msgs::Twist>("/mobile_base_controller/cmd_vel", 1);
 ros::Publisher setpoint_pub = n.advertise<std_msgs::Float64>("pmb2/pmb2/setpoint", 1);
 ros::Publisher state_pub = n.advertise<std_msgs::Float64>("pmb2/state", 1);
 ros::Subscriber sub_x = n.subscribe("pmb2/control_effort", 1, controlEffortCallback);
 setpoint.data = 0;
 ros::Subscriber action_sub = n.subscribe("robots_action/pmb2_follow", 1, pmb2_follow_cb);
 ros::Publisher feedback_action_pub = n.advertise<std_msgs::Bool>("robots_feedback/pmb2_follow", 1);

 while (ros::ok())
 {
  action = pmb2_follow.data;
  feedback = false;

  if (action == true)
  {
   if (aruco_pose.transforms.size() == 0)
   {
    cmd.linear.x = 0;
   }
   else
   {
    for (int i = 0; i < aruco_pose.transforms.size(); i++)
    {
     state.data = aruco_pose.transforms[i].transform.translation.x;
     k++;

     if (aruco_pose.transforms[i].transform.translation.x < -0.01)
     {
      cmd.linear.x = control_effort * c;
      rot_flag = true;
     }
     if (aruco_pose.transforms[i].transform.translation.x > 0.01)
     {
      rot_flag = true;
      cmd.linear.x = control_effort * c;
     }

     if (aruco_pose.transforms[i].transform.translation.x > -0.01 &&
         aruco_pose.transforms[i].transform.translation.x < 0.01)
     {
      // ROS_INFO("DONE FOLLOW");
      cmd.linear.x = 0;
      feedback = true;

      // c = 0;
     }

    }
   }

   // ROS_INFO("CMD: %f", cmd.linear.x);
   setpoint_pub.publish(setpoint);
   state_pub.publish(state);
   cmd_vel_pub.publish(cmd);
```

```
    }
    feedback_action_pub.publish(feedback);

    ros::spinOnce();
  }
}
```

ПРИЛОЖЕНИЕ В

Отправка путевых точек мобильному роботу TIAGo Base

```
#include <actionlib/client/simple_action_client.h> //actionlib
#include <move_base_msgs/MoveBaseAction.h>         //move_base action
#include <ros/ros.h>
#include <stdlib.h> //for converting argv to int
#include <tf/transform_broadcaster.h>
#include <boost/algorithm/string.hpp>
#include <fstream>
#include <list>
#include <sstream>
#include <string>
#include "std_msgs/Bool.h"

using namespace std;

typedef actionlib::SimpleActionClient<move_base_msgs::MoveBaseAction> MoveBaseClient;
int k = -1;
int p = 0;
std_msgs::Bool action, px4_feedback;

float *waypoints(int id)
{
  float *data = new float[7];
  ifstream file("/home/ros/mag_ws/src/simulation-fiduical-slam/my_pmb2/waypoints.txt");
  if (file.is_open())
  {
    string line;

    while (getline(file, line))
    {
      float waypoint_id, waypoint_x, waypoint_y, waypoint_z, waypoint_roll, waypoint_pitch, waypoint_yaw;

      istringstream iss(line);

      if (!(iss >> waypoint_id >> waypoint_x >> waypoint_y >> waypoint_z >> waypoint_roll >> waypoint_pitch >>
            waypoint_yaw))
      {
        break;
      }
      else
      {
        if (waypoint_id == id)
        {
          // ROS_INFO("YAW %f", waypoint_yaw);
          data[0] = waypoint_id;
          data[1] = waypoint_x;
```

```
        data[2] = waypoint_y;
        data[3] = waypoint_z;
        data[4] = waypoint_roll;
        data[5] = waypoint_pitch;
        data[6] = waypoint_yaw;
        return data;
       }
      }
     }
    }
   else
   {
    cout << "Don`t opened file" << endl;
   }
}

void done_cbf(const actionlib::SimpleClientGoalState &state,
            const move_base_msgs::MoveBaseResultConstPtr &result_ptr) // action done callback
{
  ROS_INFO("The action has finished.");
  ROS_INFO("DONE STATE = %s", state.toString().c_str());

  if (state == actionlib::SimpleClientGoalState::SUCCEEDED || state ==
actionlib::SimpleClientGoalState::PREEMPTED)
  {
    ROS_INFO("The robot has completed the task!");
    if (k > 21)
    {
      ros::shutdown();
    }
    else
    {
      k++;
    }
  }
  else
  {
    ROS_INFO("The robot has failed to approch the goal.");
    if (state == actionlib::SimpleClientGoalState::ABORTED)
      ROS_INFO("The mission has been aborted.");
    ros::shutdown();
  }
}

void active_cbf() // action active callback
{
  ROS_INFO("The robot starts working.");
}

void feedback_cbf(const move_base_msgs::MoveBaseFeedbackConstPtr &feedback_ptr) // action feedback
callback
{
  // ROS_INFO("The robot is at position (%.3f,%.3f)", feedback_ptr->base_position.pose.position.x,
  //          feedback_ptr->base_position.pose.position.y);
}
void action_cb(const std_msgs::BoolConstPtr &msg)
{
  action = *msg;
}
void px4_feedback_cb(const std_msgs::BoolConstPtr &msg)
{
  px4_feedback = *msg;
```

```
}
int main(int argc, char *argv[])
{
 ros::init(argc, argv, "move_base_simple_client");
 ros::NodeHandle nh;
 ros::Subscriber action_sub = nh.subscribe("robots_action/pmb2_follow", 1, action_cb);
 ros::Subscriber px4_feedback_sub = nh.subscribe("robots_feedback/px4_feedback", 1, px4_feedback_cb);
 MoveBaseClient ac("move_base", true); // action name
 actionlib::SimpleClientGoalState cnc = ac.getState();
 ROS_INFO("Wait for server start.");
 ac.waitForServer(ros::Duration(20.0)); // wait for 20 seconds
 move_base_msgs::MoveBaseGoal goal;
 while (ros::ok())
 {
  ROS_INFO("K: %d\n", k);
  float *data1 = waypoints(k);
  goal.target_pose.header.frame_id = "map";
  goal.target_pose.header.stamp = ros::Time::now();
  goal.target_pose.pose.position.x = data1[1];
  goal.target_pose.pose.position.y = data1[2];
  goal.target_pose.pose.position.z = data1[3];
  goal.target_pose.pose.orientation.x = 0;
  goal.target_pose.pose.orientation.y = 0;
  goal.target_pose.pose.orientation.z = data1[6];
  goal.target_pose.pose.orientation.w = 0.01;
  tf::Quaternion tfq;
  geometry_msgs::Quaternion msgq;
  msgq.w = 0.01;
  msgq.z = data1[6];
  tfq.setRPY(0.0, 0.0, 0.0);        // rpy to quaternion
  tf::quaternionTFToMsg(tfq, msgq); // tf quaternion to msgs quaternion
  // goal.target_pose.pose.orientation = msgq;
  ROS_INFO("Sending goal...");
  ROS_INFO("ACTION %s", action.data ? "true" : "false");
  ROS_INFO("P = %d", p);
  ROS_INFO("PX4_Fe %s", px4_feedback.data ? "true" : "false");

  // ROS_INFO("U STATE = %u", cnc.state_);
  // ROS_INFO("S STATE = %s", cnc.toString().c_str());

  if (p < 2)
  {
   ac.sendGoal(goal, done_cbf, active_cbf, feedback_cbf);
   p++;
  }
  else
  {
   if (action.data)
   {
    // ROS_INFO("STATE = %s", cnc.toString().c_str());
    ac.cancelGoal();
   }

   else if (px4_feedback.data)
   {
    ROS_WARN("GHEEEE");
    ac.sendGoal(goal, done_cbf, active_cbf, feedback_cbf); // send goal
   }
  }

  ros::spinOnce();
  sleep(2);
```

```
 }
 return 0;
}
```

ПРИЛОЖЕНИЕ Г

Узел для управления состоянием роботов

```
#include <sstream>
#include "fiducial_msgs/FiducialTransformArray.h"
#include "ros/ros.h"
#include "std_msgs/Bool.h"
#include "std_msgs/Int8.h"
std_msgs::Bool feedback_pmb2_follow, action_pmb2_follow, px4_land, px4_takeoff;
fiducial_msgs::FiducialTransformArray fiducials;
std_msgs::Int8 px4_command;
bool px4_start = true;
int last_id = 999;
int id = 0;
bool px4_action = false;
void feedback_pmb2_follow_cb(const std_msgs::BoolConstPtr &msg)
{
 feedback_pmb2_follow = *msg;
}
void fiducial_cb(const fiducial_msgs::FiducialTransformArrayConstPtr &msg)
{
 fiducials = *msg;
}
void px4_takeoff_cb(const std_msgs::BoolConstPtr &msg)
{
 px4_takeoff = *msg;
}
void px4_land_cb(const std_msgs::BoolConstPtr &msg)
{
 px4_land = *msg;
}
bool pmb2_control()
{
 ROS_INFO("PMB2_CONTROL");
 if (fiducials.transforms.size() != 0)
 {
  for (int i = 0; i < fiducials.transforms.size(); i++)
  {
   if (last_id != fiducials.transforms[i].fiducial_id)
   {
    id = fiducials.transforms[i].fiducial_id;

    action_pmb2_follow.data = true;
   }
  }
 }
 if (feedback_pmb2_follow.data == true)
 {
```

```
    last_id = id;
    action_pmb2_follow.data = false;
  }
  return feedback_pmb2_follow.data;
}
void px4_run(bool px4_action, bool px4_start)
{
  ROS_INFO("PX4 RUN");
  ROS_INFO("PX4 ACTION %s", px4_action ? "true" : "false");

  if (px4_action && !px4_takeoff.data && !px4_land.data && px4_start)
  {
    ROS_INFO("takeoff");
    px4_command.data = 1;
  }
  if (px4_takeoff.data && !px4_land.data && px4_start)
  {
    ROS_INFO("land");

    px4_command.data = 3;
  }
  if (px4_takeoff.data && px4_land.data && px4_start)
  {
    ROS_INFO("wait");

    px4_command.data = 4;
    px4_start = false;
  }
}
int main(int argc, char **argv)
{
  ros::init(argc, argv, "robots_controller");
  ros::NodeHandle n;
  ros::Publisher action_pub = n.advertise<std_msgs::Bool>("robots_action/pmb2_follow", 1);
  ros::Subscriber action_sub = n.subscribe("robots_feedback/pmb2_follow", 1, feedback_pmb2_follow_cb);
  ros::Subscriber fiducial_sub =
      n.subscribe<fiducial_msgs::FiducialTransformArray>("/fiducial_transforms", 1000, fiducial_cb);
  ros::Publisher px4_command_pub = n.advertise<std_msgs::Int8>("/pixhawk_quad/command", 1);
  ros::Subscriber takeoff_sub = n.subscribe<std_msgs::Bool>("robots_feedback/px4_takeoff", 10, px4_takeoff_cb);
  ros::Subscriber land_sub = n.subscribe<std_msgs::Bool>("robots_feedback/px4_land", 10, px4_land_cb);
  while (ros::ok())
  {
    px4_action = pmb2_control();

    if (px4_action)
    {
      px4_run(px4_action, px4_start);
      px4_start = true;
    }
    else
    {
      px4_run(px4_action, px4_start);
    }

    action_pub.publish(action_pmb2_follow);
    px4_command_pub.publish(px4_command);
    ros::spinOnce();
  }

  return 0;
}
```