МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ РОССИЙСКОЙ ФЕДЕРАЦИИ
Федеральное государственное автономное
образовательное учреждение высшего образования
«Южно-Уральский государственный университет
(национальный исследовательский университет)»

Высшая школа электроники и компьютерных наук
Кафедра «Электронные вычислительные машины»

ДОПУСТИТЬ К ЗАЩИТЕ
Заведующий кафедрой ЭВМ
_____ Д.В. Топольский
«___»_____ 2025 г.

# Разработка программного модуля для визуального обнаружения метки на объекте для выгрузки

ПОЯСНИТЕЛЬНАЯ ЗАПИСКА
К ВЫПУСКНОЙ КВАЛИФИКАЦИОННОЙ РАБОТЕ
ЮУРГУ-090301. 2025.406 ПЗ ВКР

Руководитель работы,
к.пед.н., доцент каф. ЭВМ
_____ Ю. Г. Плаксина
«___»_____2025 г.

Автор работы,
студент группы КЭ-406
_____ И. А. Маклаков
«___»_____2025 г.

Нормоконтролёр,
ст. преподавтель каф. ЭВМ
_____ С.В. Сяськов
«___»_____2025 г.

Челябинск-2025

МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ РОССИЙСКОЙ ФЕДЕРАЦИИ
Федеральное государственное автономное
образовательное учреждение высшего образования
«Южно-Уральский государственный университет
(национальный исследовательский университет)»
Высшая школа электроники и компьютерных наук
Кафедра «Электронные вычислительные машины»

УТВЕРЖДАЮ
Заведующий кафедрой ЭВМ
_____ Д.В. Топольский
«___»_____2025 г.

# ЗАДАНИЕ

**на выпускную квалификационную работу бакалавра**

студенту группы КЭ-406
Маклакову Илье Андреевичу
обучающемуся по направлению
09.03.01 «Информатика и вычислительная техника»

1. **Тема работы:** «Разработка программного модуля для визуального обнаружения метки на объекте для выгрузки» утверждена приказом по университету от «21» апреля 2025 г. № 648-13/12.

2. **Срок сдачи студентом законченной работы:** 01 июня 2025 г.

3. **Исходные данные к работе:**

3.1. Стереокамера ZED 2i.

3.2. Бортовой компьютер на базе ноутбука Machenike L15.

3.3. Симулятор на Unity.

3.4. Испытательный роботизированный мини-погрузчик.

4. **Перечень подлежащих разработке вопросов:**

4.1. Аналитический обзор научно-технической, нормативной и методической литературы по тематике работы.

4.2. Проектирование и разработка программного модуля обнаружения метки на объекте.

4.3. Внедрение программного модуля в фреймворк ROS2.

4.4. Проведение испытаний программного модуля в симуляции.

5. **Дата выдачи задания:** 2 декабря 2024 г.

Руководитель работы _____/*Ю.Г. Плаксина /*

Студент _____/*И.А. Маклаков /*

КАЛЕНДАРНЫЙ ПЛАН

| Этап | Срок сдачи | Подпись руководителя |
|---|---|---|
| Аналитический обзор научно-технической, нормативной и методической литературы по тематике работы | 01.03.2025 | |
| Проектирование и разработка программного модуля обнаружения метки на объекте | 01.04.2025 | |
| Внедрение программного модуля в фреймворк ROS2 | 01.05.2025 | |
| Проведение испытаний программного модуля в симуляции | 15.05.2025 | |
| Компоновка текста работы и сдача на нормоконтроль | 24.05.2025 | |
| Подготовка презентации и доклада | 30.05.2025 | |

Руководитель работы _____/*Ю.Г. Плаксина /*

Студент _____/*И.А. Маклаков /*

# Аннотация

Маклаков И. А. Разработка программного модуля для визуального обнаружения метки на объекте для выгрузки. – Челябинск: ЮУрГУ, ВШ ЭКН; 2025, 56 с., 25 ил., библиогр. список – 32 наим.

В данной выпускной квалификационной работе рассматривается разработка программного модуля для визуального обнаружения метки на объекте в составе мобильной робототехнической системы. Основной целью проекта является повышение точности и надёжности автоматизированной выгрузки при помощи фидуциарных маркеров и технологий компьютерного зрения. В работе используется библиотека OpenCV, фреймворк ROS2 и стек Nav2 для интеграции модуля в архитектуру управления роботом. Разработанный модуль способен в реальном времени обнаруживать ArUco-маркер, определять их позу и передавать координаты в систему навигации. Полученные результаты могут быть использованы в проектах по автоматизации складской и строительной техники, а также в других задачах, связанных с автономной робототехникой.

# ОГЛАВЛЕНИЕ

# ВВЕДЕНИЕ

Современная промышленность всё чаще обращается к использованию мобильных робототехнических систем для автоматизации различных процессов, в том числе транспортировки и разгрузки материалов. Ключевой задачей в таких системах – точное определение положения объектов для взаимодействия роботами. Одним из способов для решения этой задачи является использование визуальных меток, таких как ArUco, которые легко обнаруживаются с помощью камеры, позволяющие точно определить положение в пространстве.

Цель данной работы – разработка программного модуля для визуального обнаружения метки на объекте для выгрузки и внедрение его в робототехническую систему на базе фреймворка ROS2.

# 1. АНАЛИТИЧЕСКИЙ ОБЗОР НАУЧНО-ТЕХНИЧЕСКОЙ, НОРМАТИВНОЙ И МЕТОДИЧЕСКОЙ ЛИТЕРАТУРЫ ПО ТЕМАТИКЕ РАБОТЫ

В современном мире активно развиваются различные направления проектирования, разработки и использования робототехнических систем и комплексов. Следует отметить, что при роботизации отраслей народного хозяйства сегодня широко применяются технологии автоматизации и искусственного интеллекта. Важно подчеркнуть, что ключевым элементом любой технологии является не только уровень её готовности к внедрению и эксплуатации, но и наличие квалифицированных специалистов для её поддержки и функционирования [1].

Согласно данным, ежегодно представляемым Международной федерацией робототехники, парк промышленных роботов в мире растет из года в год [2]. В результате интенсивного научно-технического прогресса и ускорения мировой технологической гонки, связанной с растущей ролью КНР на международной экономической и технологической арене, за десять лет количество промышленных роботов в мире увеличилось почти втрое: с 1 млн в 2010 году до 3 млн в 2020 году и почти 3,5 млн в 2021 году (рисунок 1).

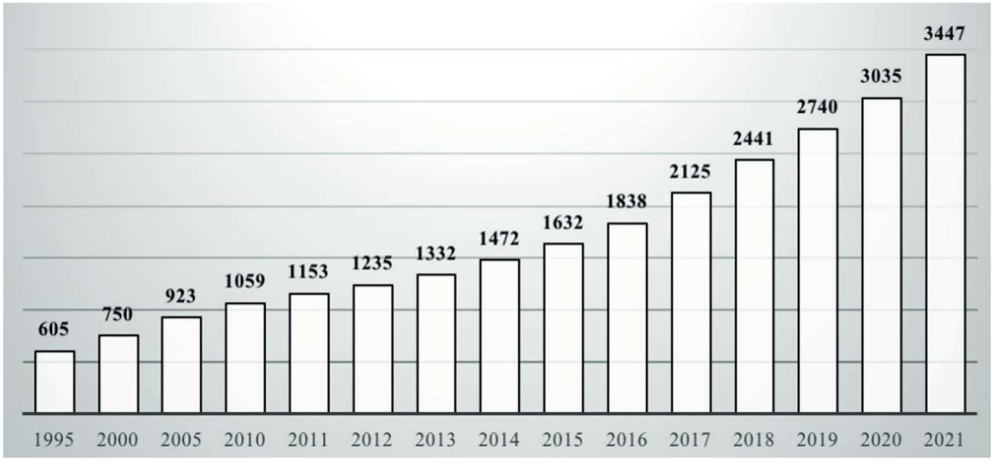


Рисунок 1 – Мировой парк промышленных роботов в Китае в 1995 – 2021 гг., тыс. шт. [2]

По этим данным можно сделать выводы, что робототехника является достаточно перспективной отраслью. Далее приведены данные, которые показывают рост устанавливаемых роботов в год в мире.

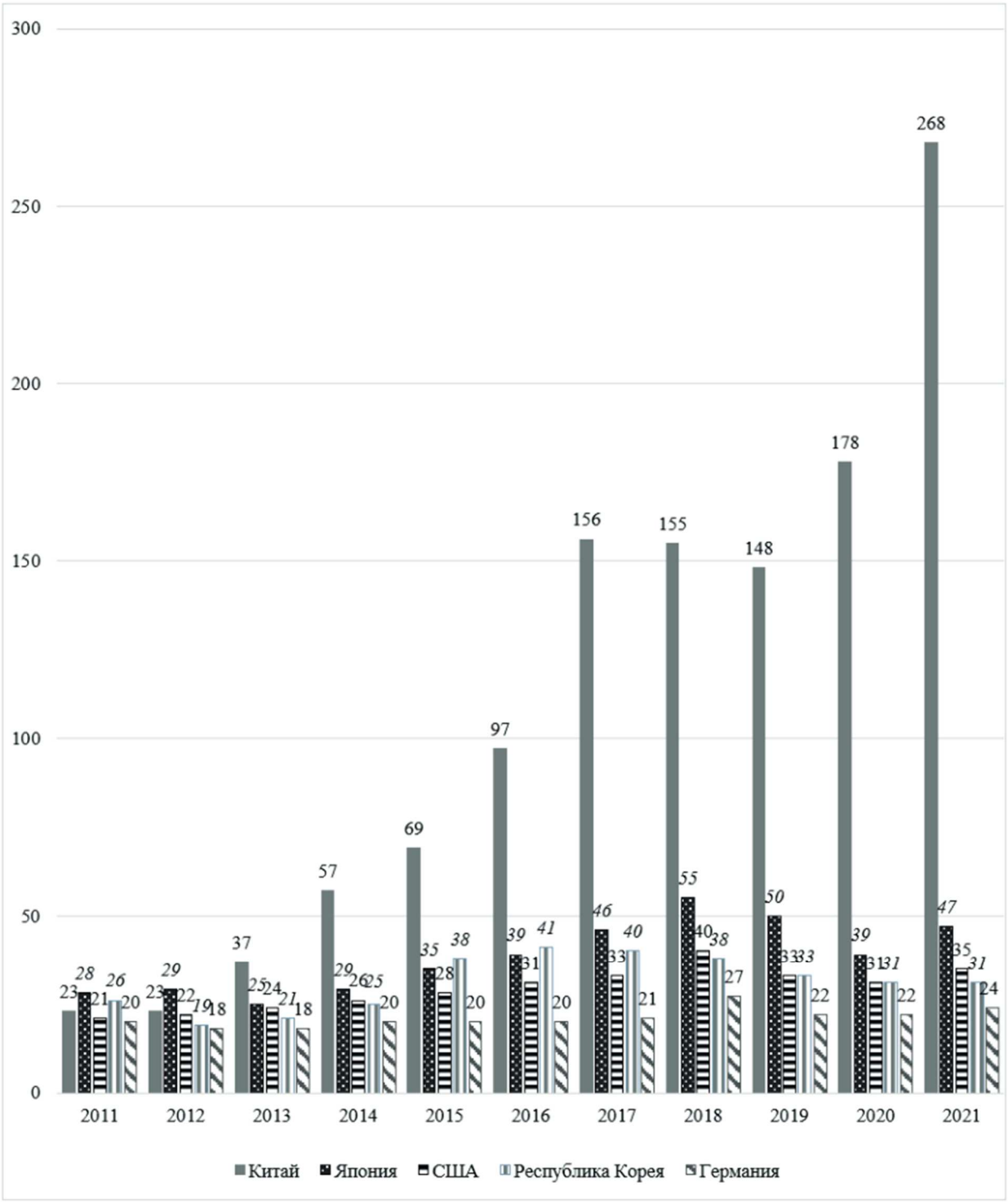


Рисунок 2 – Число устанавливаемых роботов в год в мире в отдельных странах в 2011 – 2021 гг., тыс. шт. [2]

Согласно данным рисунка 2, количество ежегодно устанавливаемых промышленных роботов демонстрирует общую тенденцию к росту. В 1995 году было установлено всего 69,3 тыс. роботов, в 2005 году их число увеличилось до 120,1 тыс., в 2015 году достигло 254 тыс., а в 2021 году составило уже 517 тыс. Начиная с 2010–2011 годов, можно говорить об устойчивом тренде на автоматизацию производства с использованием промышленных роботов.

Робот представляет собой перепрограммируемую автоматическую или автоматизированную машину, обладающую элементами антропоморфизма (человекоподобия). Это сходство может проявляться в движениях, способах решения интеллектуальных задач или общем поведении. Также существуют роботы, напоминающие животных, например, роботы-собаки-поводыри для незрячих, однако такие устройства встречаются довольно редко. В целом роботов можно разделить на две основные группы: стационарные и мобильные. Стационарные роботы имеют неподвижный корпус и выполняют движения исключительно своими рабочими органами, тогда как мобильные роботы перемещаются в пространстве вместе с корпусом.

Мобильные роботы способны перемещаться в самых разных условиях и средах, выполняя разнообразные задачи. Они могут облегчать деятельность человека, а в некоторых случаях брать на себя функции, недоступные человеку из-за ограничений его физических или интеллектуальных возможностей.

Так на рисунке 3 представлена классификация конструкций мобильных роботов.

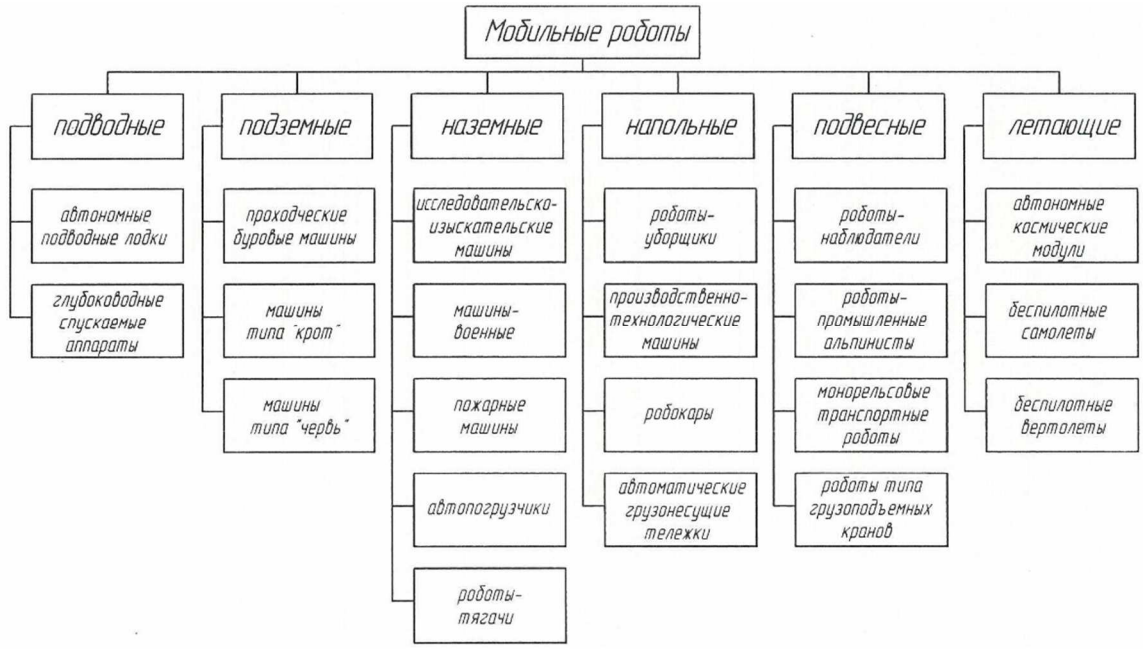


Рисунок 3 – Классификация конструкций мобильных роботов [5]

Специализация мобильного робота определяется, прежде всего, средой, в которой он должен функционировать. Исходя из особенностей среды, формулируются задачи, которые предстоит решать роботу. Одним из ключевых принципов проектирования таких машин является необходимость четко ограничить набор этих задач. Не смотря на простоту принципа, на практике его часто игнорируют, что приводит к созданию роботов с недостаточными или, наоборот, избыточными возможностями. Например, военный робот-разминировщик, который может лишь обнаруживать боеприпасы, но не обезвреживать их, что относится к недостаточным возможностям. А летающий робот-вертолет, способный не только мониторить сложные наземные устройства, но и устранять их неисправности, иллюстрирует избыточности возможностей.

Основным элементом, отличающим мобильных роботов от стационарных – это движители. Их существует достаточно много различных типов известных достаточно давно: колесные, гусеничные и шагающие рычажные. На рисунке 4 приведена схема всех известных на сегодняшний день видов движителей [5].

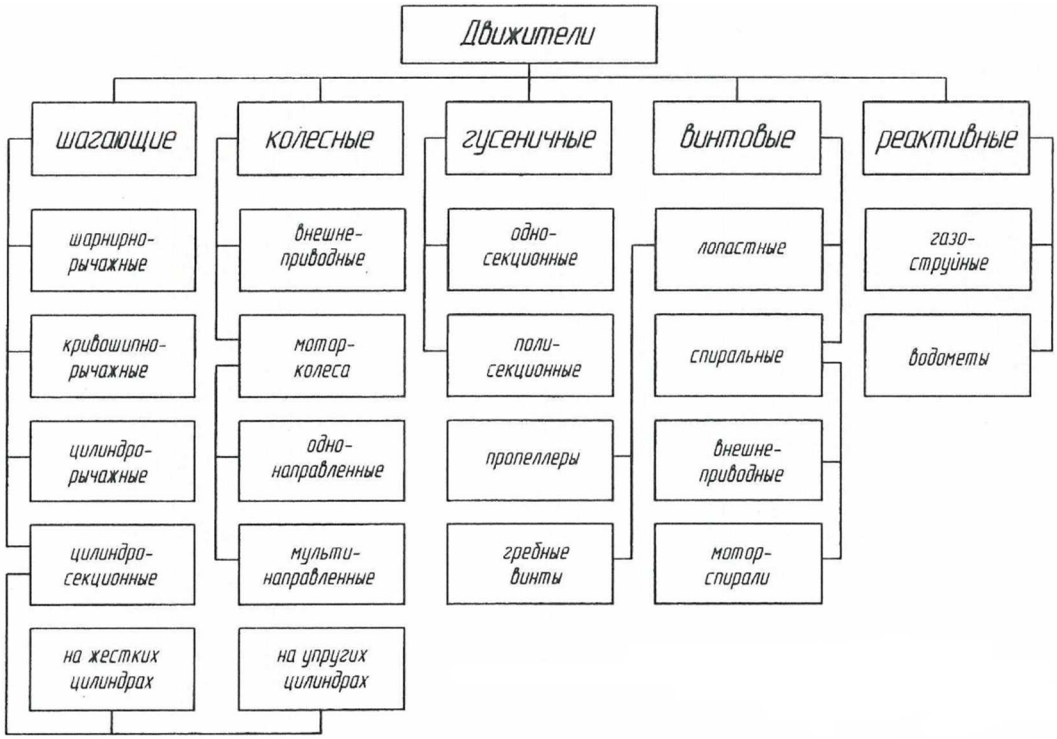


Рисунок 4 – Спецификация движителей [5]

Рассмотрим существующие решения в мире. На сегодняшний день разработкой автономными погрузчиками занимаются производители специальной техники (погрузчики, бульдозера, самосвалы, катки и прочая техника).

Caterpillar, один из мировых лидеров в производстве тяжёлой спецтехники, активно развивает направление автономного и дистанционного управления своими машинами. Технологическая платформа Cat Command обеспечивает управление техникой в режиме реального времени с помощью операторского пульта, позволяя снижать риски для персонала при работе в опасных условиях. Система поддерживает как прямое дистанционное управление, так и полуавтоматическое движение по заранее заданному маршруту, что особенно актуально для карьеров и строительных площадок. Самосвалы и погрузчики Caterpillar (рисунок 5) могут оснащаться 2D-камерами и бортовыми сенсорами, передающими данные на центральную платформу анализа, что позволяет обеспечить базовую навигацию и контроль исполнения операций [6, 7].


Рисунок 5 – Работа специальной техники Caterpillar [6]

Проект «Интеллектуальный карьер» компании БЕЛАЗ является одной из наиболее масштабных отечественных инициатив в области внедрения беспилотных транспортных средств. В рамках проекта разрабатываются карьерные самосвалы и фронтальные погрузчики, способные выполнять свои функции в автоматическом и полуавтоматическом режимах. Машины оснащаются системой дистанционного управления, а также возможностью автономного движения по маршрутам, определённым оператором или системой навигации (рисунок 6). В конструкции техники применяются модули сбора данных с камер и датчиков, позволяющие формировать картину окружающей среды и обеспечивать безопасное выполнение транспортных операций. Проект реализуется с учётом специфики условий карьеров, включая сложный рельеф, пылевые и погодные воздействия [8].


Рисунок 6 – Работа специальной техники БЕЛАЗ [8]

Volvo активно инвестирует в разработку интеллектуальной строительной техники, сочетающей электрические приводы, роботизацию и автоматическое управление. Концептуальная модель LX03, разработанная в сотрудничестве с LEGO Group, представляет собой полностью автономный электрический фронтальный погрузчик (рисунок 7). Машина оборудована системой сенсоров, включая 2D- и 3D-камеры, а также лидары, позволяющие осуществлять навигацию, избегать препятствий и адаптироваться к изменяющимся условиям среды. Уникальной особенностью конструкции является система подъема груза, имитирующая работу ножничного подъёмника, обеспечивая точность и безопасность манипуляций. Проект демонстрирует подход Volvo к созданию устойчивой и интеллектуальной строительной техники нового поколения [9].


Рисунок 7 – Фронтальный погрузчик Volvo LX03

Компания Sandvik сосредоточила усилия на создании автономных решений для подземных горных работ, где условия эксплуатации требуют особой надежности и адаптивности оборудования. Их беспилотные погрузчики предназначены для работы в ограниченных пространствах с низкой освещённостью и высокой запылённостью, что делает невозможным использование стандартных визуальных сенсоров (рисунок 8). В этих условиях техника Sandvik использует лидарные системы, инерциальные датчики и алгоритмы SLAM для точного позиционирования и построения маршрутов. Интеграция данных с различных сенсоров позволяет осуществлять плавное и безопасное перемещение по туннелям, а также повышает эффективность загрузки и транспортировки полезных ископаемых без участия оператора [10].

Рисунок 8 – Автономный робот-шахтер Sandvik

Сравним существующие решения и посмотреть, используют ли данные решения визуальные сенсоры и как с ними ведется работа.

Таблица 1 – Сравнение существующих решений

| **Компания** | **Какие функции реализует техника** | **Использование визуальных сенсоров** |
|---|---|---|
| Caterpillar | Дистанционное управление; | 2D камеры |
| БЕЛАЗ | Дистанционное управление; | 2D камеры |
| Volvo | Дистанционное управление; Автоматическое движение к точке. | 2D камеры и 3D камеры |
| Sandvik | Дистанционное управление; Автоматическое движение к точке. | Лазерные датчики и сенсоры |

Таким образом, можно обратить внимание на способы решения проблем с работой с камерами БЕЛАЗ, Caterpillar и Volvo. Первые два решения мало освящаются и подробности спрятаны за коммерческой тайной, а концепт Volvo является рекламной компанией, возможно разработки внутри данного проекта и помогут в будущем компании, но также подробной

информации катастрофически мало. Однако данные разработки показывают, что беспилотная специальная техника – это перспективная сфера, в которой ведутся активные разработки. Поэтому следует изучить вопрос работы с камерой и детектирования объектов в кадре достаточно подробно. Также следует рассмотреть распространённые методы позиционирования роботов в пространстве.

Существует множество способов анализа изображения, для поиска объекта в кадре. Так следует рассмотреть методы использующие особые точки в пространстве и векторы трехмерного потока. Особая точка определяется по ряду свойств, удовлетворяя которые, она может считаться таковой:

1. Определенность – точка должна выделяться на фоне соседних точек.
2. Устойчивость – изменение яркости, контрастности и цветовой гаммы никак не влияют на расположение точки на сцене или объекте.
3. Инвариативность – особые точки устойчивы к изменению масштаба изображения, поворота и смене ракурса.
4. Стабильность – шум на изображении не превышает определенное пороговое значение.
5. Интерпретируемость – точка представлена в формате, подходящим для дальнейшей обработки.
6. Количество – количество точек, обнаруженных в кадре, обеспечивает условие для нахождения объекта.

У каждой точки существует свое описание или же дескриптор. Дескриптор описывает особенную точку, представляя собой числовой или бинарный вектор конкретных параметров. Длину вектора определяет используемый алгоритм, а также вид его параметров [11].

Имея информацию от изображения в виде особенных точек, можно описать любое изображение для нахождения на нем определенного объекта. Однако это не дает достоверную информацию об объекте, даже при учете искривления изображения линзой камеры, существуют помехи, с которыми

достаточно тяжело бороться, такие как шум, избыточное или недостаточное(отсутствие) освещения, отражение или частицы в воздухе (пыль, снег, дождь, песок и прочее). Активные перемещения объектов или устройства также ухудшают качество объектов на изображении. Как раз для этого можно использовать особые метки, которые созданы для лучшего обнаружения при всех существующих проблемах. Специальный геометрический узор, который легко распознают камеры. Такие метки называются фидуциарные маркеры. В робототехнике используют обычно QR-код, ArUco маркер или AprilTag [12].

Так QR-код используют в навигации внутри помещения, только для прокладывания маршрута для людей. Такой метод требует достаточно четкого изображения и продолжительного времени нахождения маркера в кадре. Подобное решение реализовано внутри больницы, где пациенты и посетители могут быстро проложить маршрут от текущего местоположения, до требуемого помещения, будь то больничная палата, процедурная, регистратура или столовая. В каждом помещении расположен QR-код, отсканировав который можно проложить маршрут до следующего кабинета (рисунок 9). Такой метод достаточно неплохо себя показывает в работе с людьми, однако в роботизированных системах сложно полагаться на такой сложный вид маркера [13].

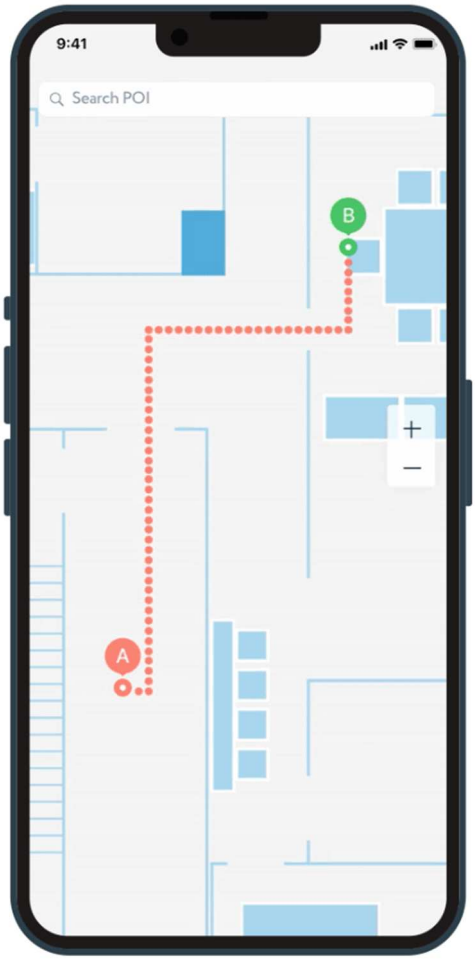


Рисунок 9 – Маршрут, построенный от отсканированного QR-кода до столовой

Фидуциарные маркеры, представляющие собой визуальные ориентиры с заданной геометрией, находят всё более широкое применение в логистических центрах и складах, особенно в системах автономной навигации. Они служат точками визуального позиционирования для мобильных роботов и дронов, обеспечивая стабильную навигацию даже в условиях недостаточной освещённости или перегруженного пространства.

Одним из ключевых преимуществ фидуциарных маркеров является их простота интеграции в существующую инфраструктуру без необходимости прокладывания физических направляющих или изменения планировки склада. Камера на роботе фиксирует маркер, и на основе распознанного

изображения система определяет точное положение устройства в пространстве. Особенно активно применяются маркеры ArUco, AprilTag.

Практические примеры демонстрируют эффективность технологии. В исследовании "Visual Fiducial Markers for Warehouse Navigation" [14] показано, как использование ArUco-маркеров позволяет снизить погрешность позиционирования до менее чем 2 см, что критично для точных операций на складе. А в решениях, применяемых компаниями вроде Amazon Robotics и Ocado, отмечается использование комбинированного подхода: фидуциарные маркеры дополняют данные от LiDAR и SLAM-систем, улучшая надёжность и адаптивность в сложных сценариях.

Amazon активно внедряет робототехнические решения для оптимизации своих складских операций. В частности, компания использует мобильных роботов, таких как Proteus (рисунок 10), которые способны автономно перемещаться по складу, избегая препятствий и взаимодействуя с другими элементами системы.

Рисунок 10 – Proteus

Для обеспечения точной навигации и позиционирования в условиях, где GPS недоступен, Amazon применяет различные методы визуальной локализации. Одним из таких методов является использование визуальных маркеров, аналогичных AprilTag, которые позволяют роботам определять своё положение относительно окружающей среды.

В ходе работы используется метод визуальной локализации, основанный на использовании карты маркеров AprilTag, созданной с помощью LiDAR-основанного SLAM, и фильтра Split Covariance Intersection Filter (Split CIF) для обработки измерений. Этот подход обеспечивает высокую точность и устойчивость локализации в условиях складских помещений.

LiDAR (Light Detection and Ranging) – это сенсор, основанный на измерении времени пролёта лазерного импульса до объекта и обратно, позволяющий строить точную трёхмерную модель окружающей среды. В контексте визуальной локализации LiDAR используется для создания карты пространства, с которой далее соотносятся данные от визуальных маркеров, таких как AprilTag. В отличие от камер, LiDAR не зависит от освещённости и может эффективно работать даже в условиях пыли и слабого освещения, что делает его особенно востребованным на складах и в промышленных помещениях [15]. SLAM (Simultaneous Localization and Mapping) – это алгоритмический подход, позволяющий роботу одновременно строить карту неизвестного пространства и определять собственное положение на этой карте. В LiDAR-ориентированной реализации SLAM алгоритм использует данные с лидаров для построения карты и трекинга перемещений, а визуальные маркеры выступают как "якоря", повышающие точность и устойчивость позиционирования. Это особенно критично в замкнутых или симметричных пространствах, где алгоритмы могут терять привязку к карте [16]. Для дальнейшего повышения точности используется Split Covariance Intersection Filter (Split CIF) – разновидность фильтрации данных, применяемая при слиянии разнородных источников информации (например, LiDAR и визуальные метки). В отличие от классического Kalman-фильтра, Split CIF не требует строгого знания корреляции между источниками и обеспечивает надёжную оценку положения даже при наличии частичных и несовпадающих данных. Это делает его эффективным решением в условиях неполной видимости или пропадания маркеров [17].

Amazon может применять аналогичные принципы в своих складах, создавая точные карты визуальных маркеров и используя их для навигации мобильных роботов. Это позволяет роботам эффективно перемещаться по складу, выполнять задачи по перемещению товаров и взаимодействовать с другими элементами системы без необходимости в дорогостоящем оборудовании на каждом роботе.

Интеграция визуальной локализации с использованием маркеров AprilTag и фильтра Split CIF может значительно повысить эффективность складских операций Amazon. Снижение времени на выполнение задач, уменьшение количества ошибок и повышение общей производительности системы способствуют улучшению обслуживания клиентов и снижению операционных затрат.

Применение методов визуальной локализации в условиях складов Amazon демонстрирует, как передовые научные разработки могут быть успешно интегрированы в реальные промышленные процессы. Это способствует повышению эффективности, снижению затрат и улучшению качества обслуживания в логистических операциях.

В свою очередь ArUco маркер широко используют в образовании, в проектах связанных с роботехникой. Открытые библиотеки для их обработки, а также простой способ их создания (маркер можно сгенерировать на специальных сайтах), позволяет достаточно легко познакомить обучающихся с принципами работы фидуциарных маркеров и особенностями их обнаружения.

Таким образом, фидуциарные маркеры представляют собой доступное и технологически зрелое решение для повышения эффективности складской логистики, особенно в условиях внедрения автономных транспортных систем.

Для решения задача больше подходят ArUco маркер и AprilTag, так как QR-код, чаще используют для передачи информации, чем для

позиционирования в пространстве из-за достаточно большого объема хранимой информации. Теперь сравним ArUco маркер и April Tag.

Эти два маркера были созданы на основе ARTAG – системе меток для отслеживания положения предметов в приложениях дополненной реальности [18]. ArUco и AprilTag создавались для увеличения надежности и точности изначальной системе меток. Чтобы окончательно определиться с выбором, ознакомимся с экспериментом тестирования на надежность обнаружения маркеров при различных углах поворота [19]. В ходе эксперимента были воссозданы идеальные условия (использование симулятора) благодаря чему исключили влияние света, шума и частиц в кадре. Использовались маркеры с одинаковым количеством пикселей для кодирования. Результаты показали большую устойчивость ArUco маркера к поворотам вокруг оси Z (рисунок 11).

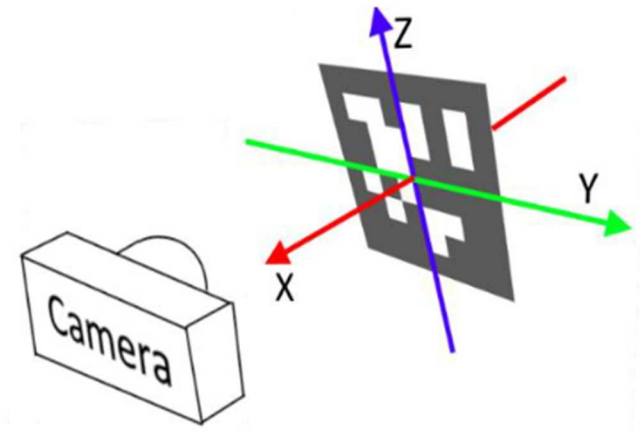


Рисунок 11 – схема вращения метки в эксперименте [19]

Подробные результаты представлены в таблице 2.

Таблица 2 – Результаты эксперимента [19]

| **Семейство и тип меток** | **Ось вращения** | |
|---|---|---|
| | X | Z |
| AprilTag – 25h7 | 99,94% | 69,96% |
| ArUco – 25h7 | 99,97% | 86,07% |

Таким образом, был выбран ArUco маркер в качестве метки для обнаружения.

## 1.2. Вывод по первой главе

Проведенный анализ литературы подтверждает значимость применения компьютерного зрения в системах автоматизированной выгрузки. Использование визуальных маркеров (ArUco, AprilTag) позволяет значительно повысить точность обнаружения и позиционирования объектов. Сравнительный анализ решений ведущих производителей показал, что внедрение таких технологий способствует увеличению эффективности мобильных робототехнических комплексов.

Оптимальным решением является комбинация автономного и ручного управления, позволяющая достичь максимальной гибкости и безопасности эксплуатации. Дальнейшее развитие данной области связано с совершенствованием алгоритмов анализа изображений и интеграцией систем машинного обучения для улучшения точности обнаружения объектов.

Таким образом, результаты работы могут быть полезны при разработке и внедрении автоматизированных систем управления мобильными роботами, ориентированными на визуальное обнаружение объектов и их обработку.

# 2. ПРОЕКТИРОВАНИЕ И РАЗРАБОТКА ПРОГРАММНОГО МОДУЛЯ ОБНАРУЖЕНИЯ МЕТКИ НА ОБЪЕКТЕ

## 2.1. Функциональные требования:

1. Загрузка изображения. Гарантирование детектирование должно достигаться на расстоянии 15 метров.
2. Определение и логирование позы (позиция и ориентация) метки. Модуль должен обрабатывать изображение для достижения наивысшей вероятности обнаружения метки и расчёта ее позиции. Далее идет логирование метки внутри модуля, для избегания повторного обнаружение, а также для передачи позы для дальнейшей работы.
3. Поддержка всех существующих видов ArUco маркеров. Модуль должен одинаково хорошо обнаруживать все существующие вариации ArUco маркеров.
4. Обработка ложных срабатываний. На этапе обнаружения должна быть проверка, которая будет гарантировать корректное обнаружение метки.

## 2.2. Нефункциональные требования:

1. Быстродействие. Должна быть поддержка работы в реальном времени (задержка меньше 100мс), обработка видеоизображения с камеры со скоростью 30 кадров в секунду.
2. Энергоэффективность. Минимальное использование вычислительных ресурсов при работе модуля.
3. Надежность. Устойчивость к загрязнениям/перекрытия маркера и изменению освещенности должна быть учтена.

Для реализации модуля воспользуемся функционалом открытой библиотекой компьютерного зрения OpenCV. Данная библиотека широко применятся в различных задачах, такие как детектирование объектов (живых и не живых), анализ видео, навигация и прочее. Все функции библиотеки

оптимизированы для высокой производительности на системах без графического сопроцессора. Данное преимущество увеличивает показатели энергоэффективности, что очень важно при использовании в беспилотных роботах, которые могут быть ограничены в электропитании. Также данная библиотека уже поддерживает работу с ArUco маркерами, что позволяет избежать этапа сборы данных для обучения нейросети, что значительно экономит время для создания прототипа. Такой подход исключает ошибок, возникших из-за неправильно подобранных обучающих данных. Также существует огромное количество функций для обработки изображения и его преобразования, такие как undisortPoints() способная на низком уровне откорректировать изображение. На рисунке 12 представлены входные и выходные данные, подмодули модуля.


Рисунок 12 – Общая архитектура модуля

Рассмотрим функционал подмодулей.

### 2.3. Подмодуль «Калибровка камеры»

Калибровка камеры выполняет одно действие – при необходимости калибрует камеру для предотвращения ошибок из-за кривизны, вызванных линзой или перспективой.

### 2.4. Подмодуль «Предобработка изображения»

Подмодуль «Предобработка изображения» подготавливает изображение с камеры для обнаружения метки в кадре. В задачи данного подмодуля входят: перевод изображения в черно-белый формат, коррекция контраста для борьбы с избыточной или недостаточной освещенностью, и обработка шумов. Архитектура подмодуля «предобработка изображения» представлена на рисунке 13


Рисунок 13 – Архитектура подмодуля предобработки изображения

### 2.5. Подмодуль «Обнаружение ArUco маркера»

В подмодуле «Обнаружение ArUco маркера» проходит детектирование квадратов и поиск изображений схожих изображений из указанного словаря для поиска определенного вида ArUco.

Последний подмодуль расчет расстояния до маркера использует позу (положение и ориентация) для расчёта расстояния до маркера. Позу можно получить уже реализованной функцией внутри библиотеки.

Модуль для обнаружения метки который будет внедрятся в фреймворк ROS.

ROS – открытый фреймворк для создания любой сложности робототехнических систем. ROS – система обмена и обработки различных сообщений от всех датчиков и устройств робота [20]. Все это достигается с помощью Node которые выполняют все эти действия.

Node – исполняемый файл, который подключён к сети ROS. Они содержат логику для решения конкретной задачи. Каждый узел связывается друг с другом посредству Topic (топик). Топик представляет из себя структуру, которая передает или принимает какой-либо тип сообщения, заранее указанный для него. Также существуют Action и Server, которые созданы для обработки предусмотренных действий [21]. Однако для понимания общей концепции организации нодов и топиков достаточно понимать их работу. На примере управления скорости электродвигателя колеса с помощью ROS показано взаимодействие двух Node, первый отвечает за работу электродвигателя колеса, а второй рассчитывает требуемую. Их «общение» реализовано посредству отправки Topic с текущей и требуемой скорости (рисунок 14).

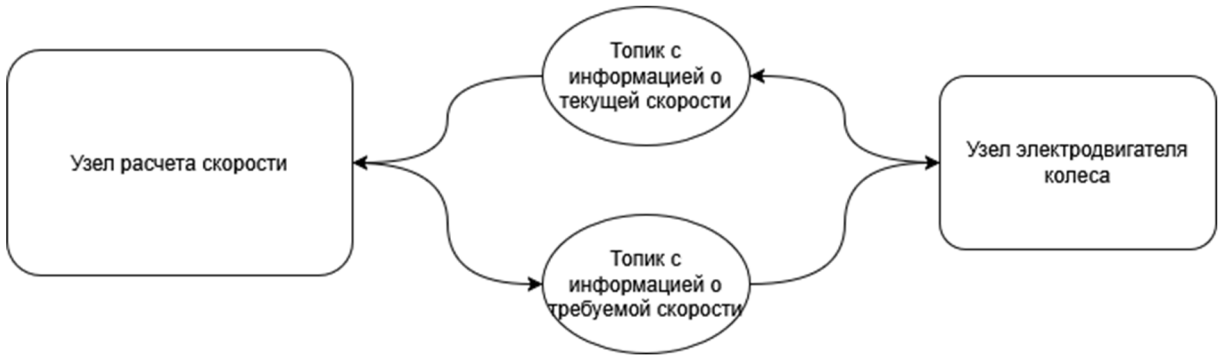


Рисунок 14 – Пример управления скорости электродвигателя колеса с помощью ROS

Важно отметить, что используется ROS2 при разработке программного модуля так как имеет значительные преимущества по отношению к ROS1.

Преимущества в архитектуре, среде обмена сообщениям, работы в реальном времени и так далее сведены в таблицу 3.

Таблица 3 – Сравнение версий ROS

| | **ROS1** | **ROS2** |
|---|---|---|
| Архитектура | Использует централизованный мастер (ROS Master) для управления коммуникацией между узлами, что создаёт единую точку отказа. | Отказывается от мастера, применяя децентрализованную архитектуру с использованием DDS (Data Distribution Service), обеспечивая более надёжное взаимодействие между узлами без единой точки отказа. |
| Среда обмена сообщениями | Применяет собственный протокол на основе TCP/ROS и UDP/ROS, ограниченный в надёжности и реальном времени. | Использует стандарт DDS, обеспечивающий высокую производительность, низкую задержку и возможности настройки QoS (Quality of Service). |
| Реальное время | Не предназначен для задач реального времени; возможны задержки в выполнении. | Улучшенная поддержка реального времени благодаря использованию DDS и многопоточной архитектуры. |
| Безопасность | Отсутствуют встроенные механизмы безопасности; данные передаются в незашифрованном виде. | Интегрированы функции безопасности, включая шифрование, аутентификацию и контроль доступа, благодаря использованию DDS. |
| Кроссплат-форменность | Официально поддерживается только на Linux; ограниченная поддержка других ОС. | Поддерживает Linux, Windows и macOS, что облегчает разработку на различных платформах. |
| Система сборки | Использует catkin; поддерживает неразделённую сборку нескольких пакетов в одном контексте CMake. | Переход на ament; поддерживает изолированную сборку пакетов, улучшая модульность и управление зависимостями. |

Продолжение таблицы 3

| | **ROS1** | **ROS2** |
|---|---|---|
| Параметры | Управляются через глобальный сервер параметров, связанный с ROS Master. | Каждый узел управляет своими параметрами локально, повышая гибкость и надёжность. |
| Языки программирования | Библиотеки roscpp для C++ и rospy для Python разрабатывались отдельно, что приводило к различиям в функционале. | Введена общая клиентская библиотека rcl на C, поверх которой строятся интерфейсы для различных языков, обеспечивая единообразие и упрощая добавление новых языков. |

Таким образом, выбор ROS2 обусловлен улучшенной архитектурой, которая более отказоустойчив, поддержкой полноценной работы в реальном времени, а также повышенной безопасностью, чем в предыдущей версии.

Рассмотрим внедрение в инструмент для реализации автономного движения робота Navigation Stack 2 (Nav2) [22].

Для реализации навигации робота применяется стек. Стек – это набор инструментов, языков программирования, фреймворков и библиотек, используемых при разработке. Nav2 широко применяется, так как имеет исчерпывающий функционал для различных настроек, как уже существующих, так и для разработки собственных инструментов.

Стек полностью описывает весь цикл для реализации автономного движения. Основными элементами Nav2 являются:

1. Planner Server (Глобальный планировщик). Отвечает за построение глобального пути от текущего положения робота до заданной цели, используя карту окружающей среды. Часто применяются алгоритмы A* или Dijkstra для расчёта оптимального маршрута.
2. Controller Server (Локальный контроллер). Использует локальные данные сенсоров для генерации команд управления, обеспечивая следование робота по глобальному пути с учётом динамических

препятствий. Примеры контроллеров: Regulated Pure Pursuit, DWB Controller.

3. Behavior Tree (BT) Navigator. Управляет последовательностью действий робота с помощью дерева поведения, позволяя гибко настраивать логику навигации и реакции на различные события.
4. Lifecycle Manager. Обеспечивает управление жизненным циклом узлов Nav2, включая их инициализацию, активацию и завершение работы, что способствует стабильности и надёжности системы.
5. Costmaps (Карты стоимости). Создают представление окружающей среды в виде сетки, где каждой ячейке присваивается "стоимость" прохождения. Это помогает планировщику избегать препятствий и выбирать безопасные маршруты.
6. Recovery Behaviors (Поведения восстановления). Определяют действия, предпринимаемые роботом в случае возникновения проблем, таких как застревание или потеря пути, например, отъезд назад или повторное планирование маршрута.
7. Smoother Server. Улучшает плавность траектории, сглаживая резкие повороты и обеспечивая более естественное движение робота.
8. Waypoint Follower. Позволяет роботу следовать по заранее заданным точкам (waypoints), что полезно для патрулирования или выполнения маршрутов.
9. Collision Monitor. Отслеживает возможные столкновения и принимает меры для их предотвращения, повышая безопасность навигации.
10. Velocity Smoother. Сглаживает команды скорости, предотвращая резкие ускорения или торможения, что способствует более стабильному движению.

Так в ходе движения погрузчика до указанной конечной точки можно обрабатывать возникающие или заранее известные (Costmap) препятствия, способы решения (Planner Server, Controller Server, Recovery Behaviors), реализовывать дополнительные действия, такие как сбор данных с датчиков,

управление навесным оборудованием (Waypoint Follower), что позволяет создавать сценарии для каждого вида задач (BT Navigator) (рисунок 15).

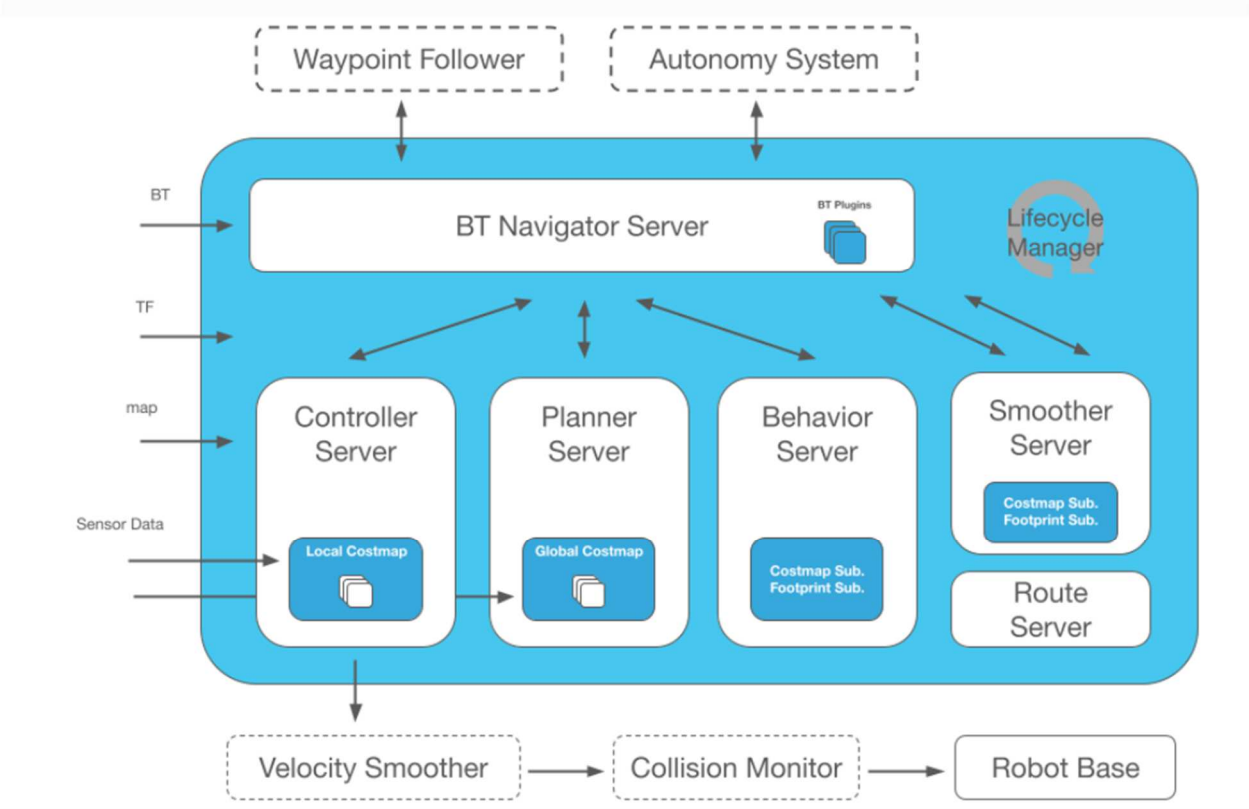


Рисунок 15 – Архитектура Nav2 [23]

При разработке программного модуля используется инструмент, отвечающий за поведение (дерево поведений (Behavior Tree) и сервер поведений (Behavior Server)), которое используется при получении точки.

Nav2 использует дерево поведений для описания поведения при следовании к цели. Дерево поведений – структура данных, используемая для описания поведения в виде дерева, где каждый узел представляет собой какое-либо действие или задачу. Данный метод применяется для создания искусственного интеллекта во многих сферах (робототехника, симуляторы видеоигры). Такой способ создания действий позволяет реализовать структуру поведения любой сложности и размеров. Структура дерева поведения легко расширяются за счет модульности и иерархии в виде дерева [24].

Основные элементы деревьев поведений делятся на два типа: узел потока и узел исполнитель. К узлам потока относят:

1. Последовательность (Sequence). Является родительским узлом для узлов исполнителей. Имеют три состояния успех (Success), выполнение (Running) и провал (Failure). Дерево поведений имеет некоторую частоту исполнения (тик), в момент которой узел выполняет свое действие. В первый тик узел возвращает состояние выполнения. В зависимости от результата возвращает успех или провал соответственно. Последовательность всегда должна иметь дочерний узел и в зависимости от его результата возвращает значение. Пока дочерний узел выполняется последовательность тоже будет возвращать выполнение, также как и при провале или успехе. Если дочерних узлов несколько, то возвращает выполнение пока не получит статус успеха от последнего узла, либо статус провала.
2. Запасной вариант (Fallback). В отличии от последовательности возвращает успех если хотя бы один из дочерних узлов завершился со статусом успех. Данный узел создан для решения возможных ошибок в поведении.

К узлам действий относят:

1. Действия (Action). Данный узел содержит в себе обработку какого-либо действия, по результат которого возвращает статус. Например: открой дверь, подними руку, включи свет.
2. Условие (Condition). Данный узел проверяет какое-либо условие, возвращает только статус успеха или провала. Например: дверь открыта, рука исправна, есть ли электричество.
3. Декоратор (Decorator). Является уникальным узлом действия так, как его действия полностью зависят от дочернего узла, благодаря чему можно описать огромное количество действий [25].

При разработке программного модуля наиболее подходит узел условия, который будет проверять наличие ArUco маркера на изображении, для

дальнейшего подъезда к нему. Таким образом, программного модуль является начальным узлом в дереве поведений для выгрузки содержимого ковша, помещённым внутри Fallback для обработки случая, когда маркера не будет в кадре камеры (рисунок 16).

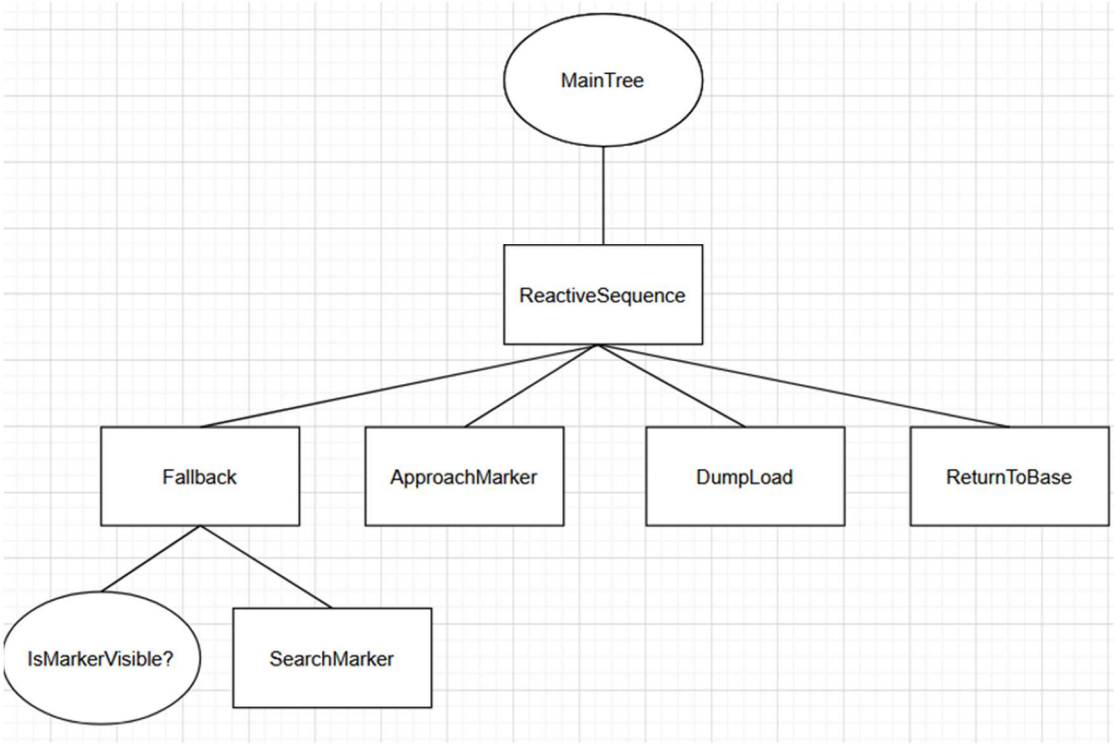


Рисунок 16 – Дерево поведений для разгрузки погрузчика

В процессе разработки программного модуля для обнаружения метки на объекте для выгрузки, основным компонентом является модуль зрения – камера, способная распознавать визуальные маркеры, такие как ArUco, AprilTag или QR-коды. Эти маркеры служат ориентирами для локализации, калибровки или навигации, и потому надёжность системы распознавания критична. Однако тестирование такого модуля в реальном мире требует много времени и ресурсов. Игровые движки становятся удобной и мощной средой для симуляции камер, сцены и поведения маркеров в различных условиях.

Среди игровых движков Unity – один из наиболее популярных в робототехнических проектах. Его широкие графические возможности, гибкость в настройке сцен и поддержка виртуальных камер позволяют точно

моделировать изображение, поступающее на вход модуля распознавания. Несмотря на то, что Unity изначально не создавался для задач робототехники, он хорошо интегрируется с ROS с помощью специальных мостов (дополнительное расширение), что позволяет ему работать как часть полноценной робототехнической системы. Маркеры можно размещать прямо в 3D-пространстве как объекты со своими текстурами, а симулируемая камера может передавать изображение в OpenCV или другие библиотеки.

Unreal Engine представляет собой высокопроизводительный графический движок, ориентированный на фотореалистичную визуализацию сцен, что делает его особенно актуальным для задач, связанных с тестированием алгоритмов компьютерного зрения в условиях, приближённых к реальным. Благодаря точному моделированию освещения, отражений, теней и других оптических эффектов, Unreal позволяет достоверно воспроизводить сценарии, в которых маркеры могут быть частично засвечены, искажены или находиться в условиях сложного освещения. Это критически важно при верификации устойчивости алгоритмов распознавания маркеров, чувствительных к внешним условиям съёмки. Вместе с тем, интеграция Unreal Engine в робототехнический стек, в частности в системы на базе ROS, требует дополнительных усилий и, как правило, реализуется с использованием сторонних мостов или пользовательских решений, что увеличивает порог входа и затраты на настройку, по сравнению с Unity.

В экосистеме ROS наиболее традиционным выбором остаётся симулятор Gazebo, и его новое поколение – Ignition. Эти симуляторы ориентированы в первую очередь на физическую достоверность поведения роботов, и в них встроены базовые средства визуализации. Камеры в Gazebo могут транслировать изображение в ROS-топики, что удобно для дальнейшей обработки. Хотя качество графики здесь уступает Unity или Unreal, Gazebo обеспечивает простую реализацию сценариев, где важна не столько визуальная реалистичность, сколько совместимость с робототехническим

стеком и возможность быстро запускать десятки итераций в автоматическом режиме.

Среди специализированных симуляторов для робототехники выделяется Webots. Он прост в освоении и предназначен в первую очередь для образовательных и исследовательских задач. Несмотря на то, что он использует более упрощённую графику, Webots позволяет легко моделировать камеры и размещать маркеры в сцене. Обработка изображения может происходить внутри движка или через внешние библиотеки, и платформа хорошо интегрируется с ROS и Python.

Отдельно стоит упомянуть Isaac Sim – мощный симулятор от NVIDIA, построенный на Omniverse. Он ориентирован на задачи ИИ и цифровых двойников и предлагает средства для физически точного моделирования камер, включая настройки объектива, искажения, шумы, а также сложные сценарии освещения. Эта платформа особенно интересна для создания синтетических наборов данных и обучения моделей детекции в условиях, приближенных к реальным. Однако она предъявляет высокие требования к оборудованию и требует времени на изучение.

Таким образом, выбор симуляционной платформы для тестирования модулей компьютерного зрения, в частности распознавания маркеров, должен учитывать не только графические возможности движка, но и степень его интеграции с инструментами робототехнической разработки, поддерживаемые модели камер, параметры визуализации и возможности автоматизации экспериментов. Ниже представлена сравнительная таблица 4, в которой систематизированы ключевые характеристики наиболее распространённых симуляционных сред, применяемых для моделирования камер и анализа устойчивости алгоритмов детекции маркеров в виртуальных условиях.

Таблица4 – Сравнение игровых движков для испытания программного модуля

| | **Unity** | **Unreal Engine** | **Gazebo** | **Webots** | **Isaac Sim** |
|---|---|---|---|---|---|
| Виды камер | Перспек-тивная, ортогра-фическая | Физическая модель камеры | Простая перспек-тивная, | Перспек-тивная | Физическая модель камеры |
| Симуля-ция шумов | С помощью шейдеров или скриптов | Настройки внутри камеры | Базовые эффекты внутри камеры | Базовые эффекты внутри камеры | Настройки внутри камеры |
| Мак-сималь-ное разре-шение камеры | 4K | 8K | 1080p | 4K | 4K |
| Макси-мальная частота кадров камеры | 120FPS (Frame per second, кадры в секунду) | 120FPS | 30FPS | 30FPS | 120FPS |
| Инте-грация ROS | С помощью готового расширения (ROS bridge) | Написание собствен-ного решения | Нативная поддержка | С помощью готового решения (ROS2 Interface) | С помощью готового решения внутри Isaac SDK |

Продолжение таблицы 4

| | | | | | |
|---|---|---|---|---|---|
| Инте-грация OpenCV | С помощью готового расширения (Python bridge) | С помощью готового расширения (ROS Wrapper) | Нативная поддержка | С помощью готового решения (Python API) | С помощью готового расширения (Omniverse Streaming) |
| Реализм сцены | Высокий | Фоторе-ализм | Средний | Средний | Фоторе-ализм |

## 2.6. Вывод по второй главе

Результат второй главы – функциональные и нефункциональные требования для создания модуля, построена схема архитектуры модуля с подробным описанием работы каждой функции, представлен способ решения с помощью библиотеки OpenCV, рассмотрен фреймворк ROS2 и план внедрения в стек Nav2 в виде узла условия внутри основного дерева поведения для выгрузки.

# 3. ВНЕДРЕНИЕ ПРОГРАММНОГО МОДУЛЯ В ФРЕЙМВОРК ROS2

## 3.1. Средства реализации

Для реализации программного модуля был использован фреймворк ROS2 в связке со стеком NAV2, текстовый редактор Visual Studio Code, язык программирования C++ с менеджером сборки colcon и открытой библиотеки для компьютерного зрения OpenCV. Инструмент сборки colcon позволяет быстро собирать проекты, автоматически генерируя зависимости и поддерживая различные системы. Каждый пакет ROS2 состоит из файлов конфигурации, заголовочных файлов, исходного кода и системных зависимостей. Для облегчения сборки используются два файла. Первый package.xml содержит базовые сведения, такие как используемые библиотеки, систему сборки (C++ или Python), информацию о типе и версии сборки. Файл setup.py содержит и устанавливает зависимости, а также способы обращения к ним.

Разработка на фреймворке ROS2 использует принципы объектно-ориентированного программирования. Так каждая узел (Node) создается с определенной целью, функционал Node реализуется в обработке изображения. Такой подход помогает легко масштабировать и отлаживать модуль. Таким образом, будет создан Node «aruco_detector» в котором будет проходить основная работа.

Для Node нужно принимать данные с камеры, как сам видеопоток (цветовая модель, разрешение изображения, поток байт и прочее), так и параметры камеры, такие как матрица камеры, коэффициенты искажения. Все эти параметры камера уже отправляет в систему ROS поэтому остается только «подписать» данные к узлу «aruco_detector». Это реализовывается при помощи метода create_subscription<T>(), который создает объект «подписчика». Данный объект служит для принятия сообщения из указанного в параметрах метода и обрабатывается в отдельной функции.

Таким образом, принимается всю необходимую информацию об изображении и параметрах камеры из Topic /image и /image/camera_info. Выполнение функций для обработки нескольких «подписчиков» необходимо поместить в одну группу callback_group для избежания «гонки» между двумя обработчиками «подписчиков».

Далее для работы с OpenCV необходимо перевести тип sensor_msgs::msg::Image в тип cv::Mat для дальнейшей обработки изображения с помощью OpenCV. Для этого используется cv_bridge (мост) пакет для конвертации изображения из ROS типа в OpenCV и обратно методы cv_bridge::toCvShare() и cv_bridge::CvImage() соответственно.

Далее с помощью OpenCV определяем метку на изображение с помощью метода detectMarkers(). Внутри данного метода происходят следующие действия:

1. Предобработка изображения. Перевод в черно-белое изображение, так как ArUco маркеры черно-белые.
2. Поиск потенциальных контуров. Обнаружение контуров для поиска мест с резким контрастом, в которых потенциально может быть маркер.
3. Поиск областей квадратов. Среди мест с резким контрастом находятся квадраты.
4. Проверки для каждого маркера. Определяют какому словарю (набор уникальных бинарных квадратных маркеров с личным ID) принадлежат ArUco маркер, а также является ли квадрат маркером.
5. Идентификация. После проверки квадрата, при соблюдении всех параметров, записывается его ID.
6. Корректировка углов и точек. Определяется место в виде 4 углов и записывается в список углов.
7. Выдача результата. После успешного выполнения метода выдает два списка: список ID номеров маркеров и список углов каждого маркера.

После получения двух списков, нужно соотнести расположение маркера в 2D пространстве и перенести его в 3D. Необходимо учитывать искривление линзы камеры, которое искажает (деформирует) изображение (рисунок 17) [26].

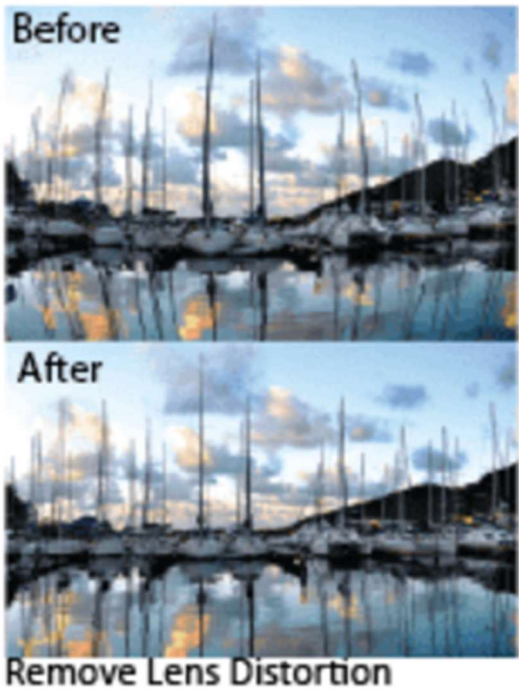


Рисунок 17 – Искажение линзой камеры [26]

Для исправления искажение камеры необходимо использовать метод undistortPoints(). Данный метод исправляет искажения 2D-точки, используя параметры камеры (матрица камеры и коэффициенты искажения). На выходе получаем исправленные точки.

Для 3D-реконструкции используется метод estimatePoseSingleMarkers(). 3D-реконструкция – процесс восстановления координат или трехмерной формы на основе двухмерной проекции [27].

Данный метод получает список углов каждого маркера, длину стороны маркера в метрах, матрицу камеры и коэффициенты искажений. На выходе формируется вектор поворота (вектор ориентации) и вектор трансляции (вектор положения) метки в 3D пространстве.

Теперь имея позу (вектор поворота и вектор трансляции) необходимо передать ее в тип для ROS2. Так как по маркеру определяется положение

объекта выгрузки. Для этого необходимо перевести углы Эйлера [28], записанные в векторе поворота, в кватернионы [29], так как тип данных, отвечающий за позу в ROS2, использует их. Для преобразования вектора поворота в кватернион выполним следующие шаги:

1. Нормирование вектора поворота. С помощью метода OpenCV norm() нормируем вектор поворота.
2. Нахождение осей вращения. Значение каждого вектора делим на результат метода norm().
3. Получение кватерниона. С помощью библиотеки ROS2 отвечающей за преобразование координат между системами координат, создадим переменную хранящую в себе полученный кватернион, используя метод setRotation().

Однако полученная точка просчитана в системе координат относительно камеры, что невозможно использовать при навигации. Для этого нужно пересчитать позу относительно фрейма map.

Фрейм – это система координат, относительно которого описывается положение объектов. Информация о фреймах содержится в топике /tf которое, в свою очередь, формирует дерево из всех фреймов и их отношение друг к другу (рисунок 18).

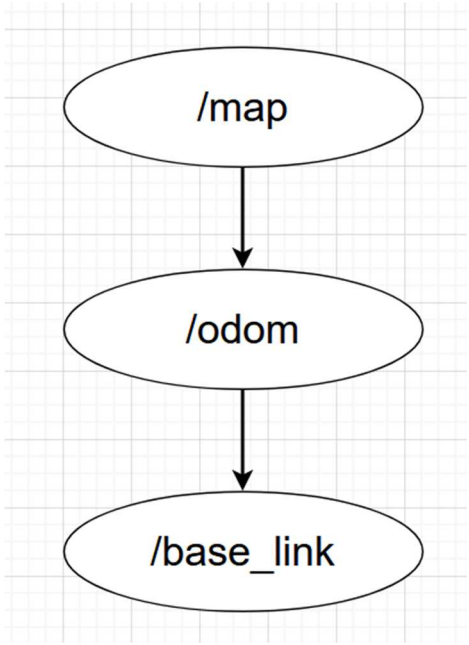


Рисунок 18 – Дерево /tf по стандарту REP-105 [30]

По стандарту REP-105 существуют три обязательных фрейма:

1. Map. Глобальная система координат (фиксированное). Определяет абсолютное положение робота в мире.
2. Odom. Локальная система координат. Отслеживает перемещение робота относительно начальной точки (map). Рассчитывается с помощью датчиков (IMU, энкодеры, GPS, камеры).
3. Base_link. Центр робота относительно осей вращения колес. Является родительским фреймом для различных датчиков и устройств робота.

Получены координаты относительно фрейма камеры, из-за чего сложно понять, где находится маркер относительно машины. Для точного позиционирования маркера необходимо пересчитать координаты в систему координат map с помощью метода библиотеки tf2 (библиотека ROS2 для работы с фреймами) doTransform(), который пересчитает координаты в отношении необходимого фрейма.

Схема взаимодействия камеры и «aruco_detector» представлена на рисунке 19.

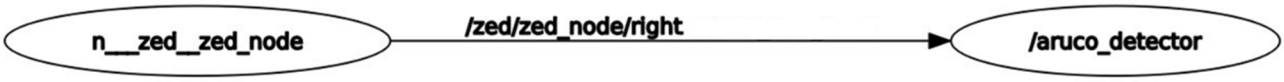


Рисунок 19 – Схема взаимодействия камеры и модуля (rqt_graph)

Модуль получает с камеры (n__zed_zed_node) сообщение /zed/zed_node/right в котором лежит поток типов данных sensor_msg/Image и sensor_msg/Camera_Info, содержащие необходимую информацию для работы «aruco_detector». Итоговая архитектура модуля показана на рисунке 20.

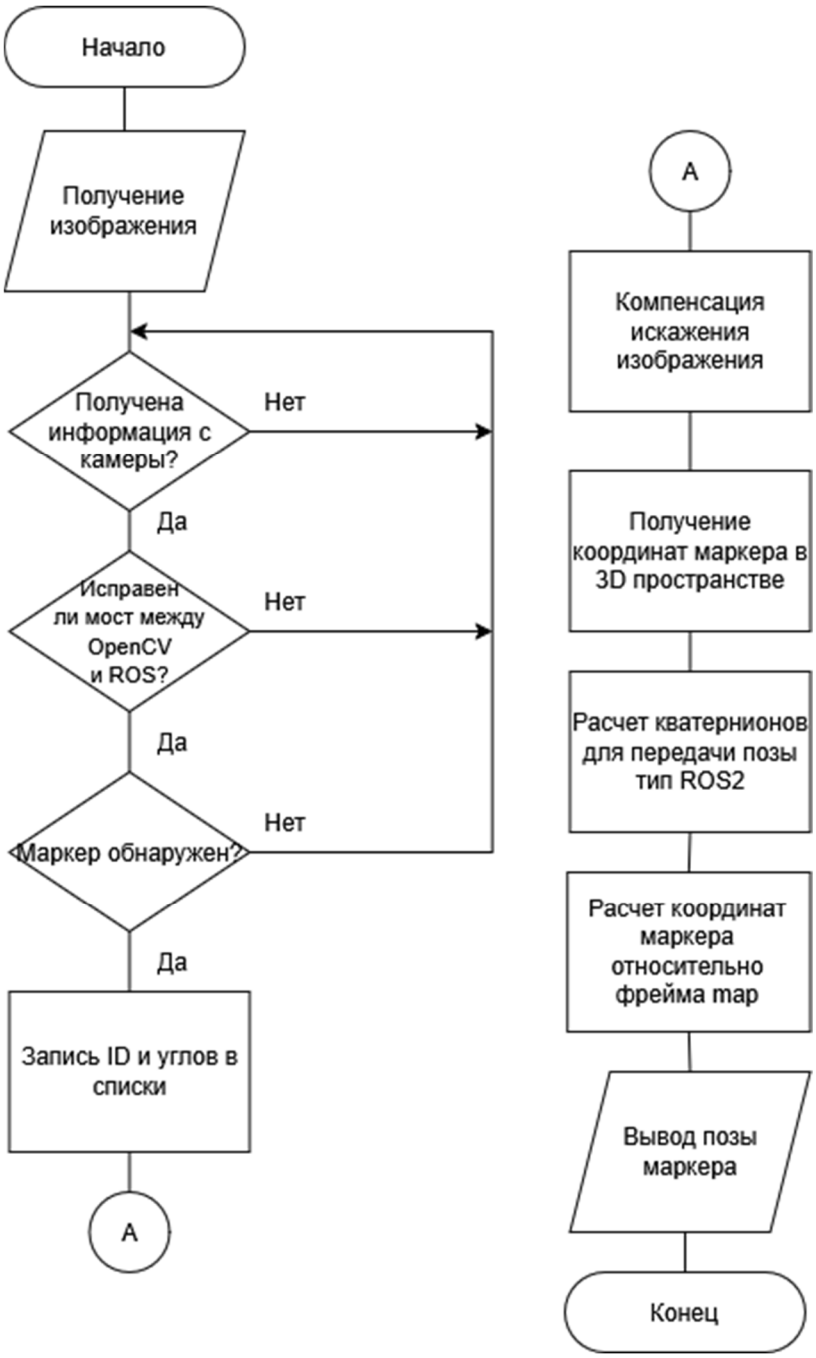


Рисунок 20 – Графическая схема алгоритма программного модуля

Так как узел «aruco_detector» необходимо включить в дерево поведений требуется «завернуть» его работу в одно из действий для дерева поведения. Для реализации программного модуля целесообразно использовать Condition, так как пока маркер не обнаружен (не выполнено условие) выполнение дальнейших действий невозможно.

Каждый элемент дерева поведений проверяется в определенный момент времени, так называемый tick. Для построения элемента можно

воспользоваться документацией [31]. В ней сказано, что для реализации элемента дерева необходим сам конструктор класса и порты ввода/вывода информации. Также для реализации Condition необходим метод tick(). Таким образом, всю реализацию «aruco_detector» можно поместить в этот метод. Если маркер в ходе работы был обнаружен, то Condition должен вернуться в статус SUCCESS, во всех иных ситуациях FAILURE (рисунок 21). В случае положительного статуса (SUCCESS), дерево продолжает работу (находится в статусе RUNNING), в противном случае останавливается все дерево (переход в состояние FAILURE).


Рисунок 21 – Работа aruco_detector в качестве узла Condition

## 3.2. Вывод по третьей главе

В рамках данной главы был реализован программный модуль для визуального обнаружения метки и успешно внедрён во фреймворк ROS2. Обеспечение стабильного и точного определения положения метки в

пространстве, что удалось реализовать с помощью библиотеки OpenCV и средств ROS2.

Модуль работает в реальном времени, получает данные с камеры, обрабатывает изображение, находит ArUco-маркер и определяет его позу. Полученные координаты переводятся из системы камеры в глобальную систему координат, что позволяет использовать их для дальнейшей навигации робота. Также модуль встроен в стек Nav2 в виде узла дерева поведения, что позволяет управлять действиями погрузчика в зависимости от наличия метки в кадре.

Работа с ROS2 и Nav2 показала свою актуальность и удобство при построении современных решений в области мобильной робототехники.

# 4. ПРОВЕДЕНИЕ ИСПЫТАНИЙ ПРОГРАММНОГО МОДУЛЯ В СИМУЛЯЦИИ

В качестве «полигона» для испытаний выбран игровой движок Unity [32]. Данное программное обеспечение технически гибкое позволяющее разрабатывать мультиплатформенные симуляторы, также в достаточной степени эмулирует поведение теней и света, что дает более реалистичные результаты испытаний в обнаружении ArUco. Существующее расширение для работы с ROS2, позволяет эмулировать поведение реальных роботов (такое же общение топиками, как у реальных роботов). В сравнении более известными симуляторами для роботов, таких как Gazebo или Webots, обладает низким порогом для вхождения при разработке двойников роботов. Также выбор обусловлен тем, что в данном программном обеспечении уже разработана цифровая копия погрузчика со всеми сенсорами и датчиками, а также набор различных карт для испытаний.

Для проверки создан маркер размером 30х30 см из словаря 6х6 ID 24 (рисунок 22).

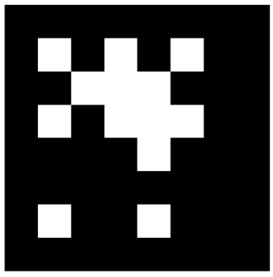

Рисунок 22 – ArUco маркер из словаря 6х6 ID24

Данный маркер расположили на сцене и провели следующие испытания:

1. Обнаружение маркера.
2. Максимальная дистанция детектирования.
3. Точность измерения дистанции.
4. Испытание максимальных углов обнаружения.

### 4.1. Первое испытание

В данном испытании проверяется работоспособность программного модуля. При обнаружении в консоли должны увидеть сообщение об успешном детектировании и параметры маркера (поза и ID) и получение изображения с камеры, где явно видно обнаруженный маркер. По рисунку 23 видно, что испытание успешно пройдено.

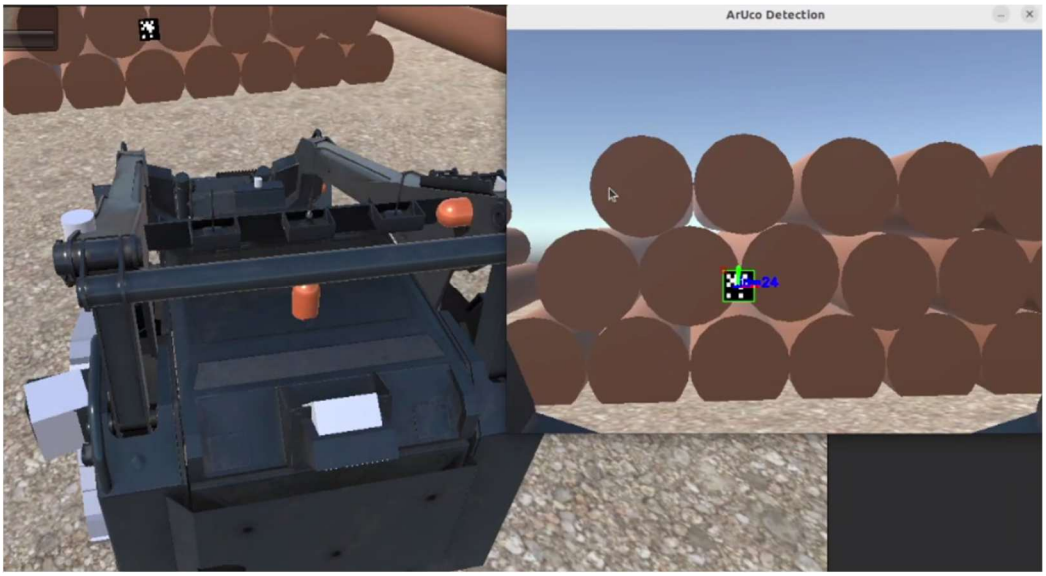


Рисунок 23 – Первое испытание

### 4.2. Второе испытание

Для второго испытания был создан специальный полигон с сеткой для измерения расстояния, где каждый квадрат на ней равен 25x25см.

В ходе испытаний было выяснено, что при разрешении 640x320 максимальная дистанция составила 15 метров. Результат удовлетворяет требования ТЗ, однако нельзя забывать, что в симуляции идеальные условия из-за чего результат в реальности может отличаться. На рисунке 24 красной линией показано расстояние от камеры (левой машины) до маркера (на правой машине).

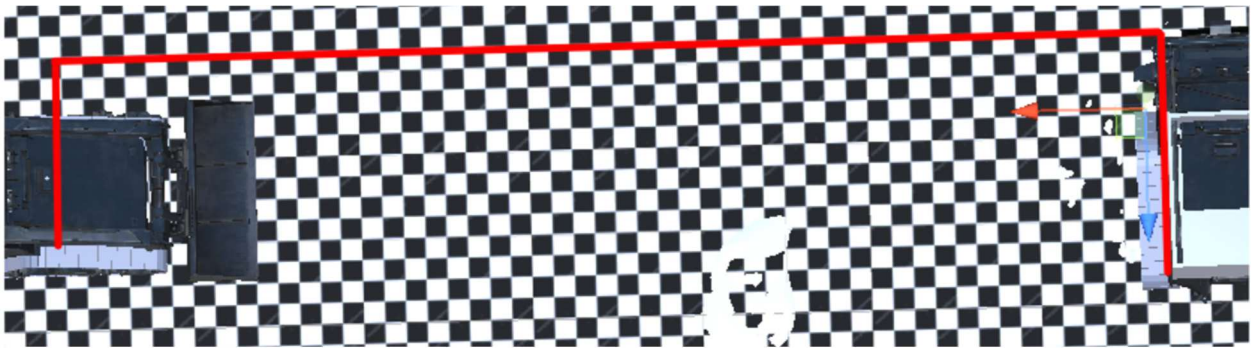

Рисунок 24 – Второе испытание

### 4.3. Третье испытание

В данном испытании использовался второй полигон. Проведена серия испытаний от 3 до 15 метров с шагом в 2 метра. По результатам испытания составлена таблица 5.

Таблица 5 – Третье испытание

| Эталонное значение, м | Полученное значение, м | Абсолютная погрешность, м | Относительная погрешность, % |
|---|---|---|---|
| 3 | 3,095 | +0,095 | 3,17 |
| 5 | 4,985 | -0,015 | 0,30 |
| 7 | 6,989 | -0,011 | 0,16 |
| 9 | 8,957 | -0,043 | 0,48 |
| 11 | 10,967 | -0,033 | 0,30 |
| 13 | 13,034 | +0,034 | 0,26 |
| 15 | 14,928 | -0,072 | 0,48 |

Средняя абсолютная погрешность оказалась 0,043м, а средняя относительная погрешность 0,736%. Данные значения удовлетворяют потребности, так как ошибка в 5 см относительно размеров погрузчика, незначительна.

### 4.4. Четвертое испытание

Заключительным испытанием является выявление максимальных углов тангажа (ось y) и рысканья (ось z). Крен нас не интересует так как при вращении вокруг оси x перекрытия маркера не возникает (рисунок 25).

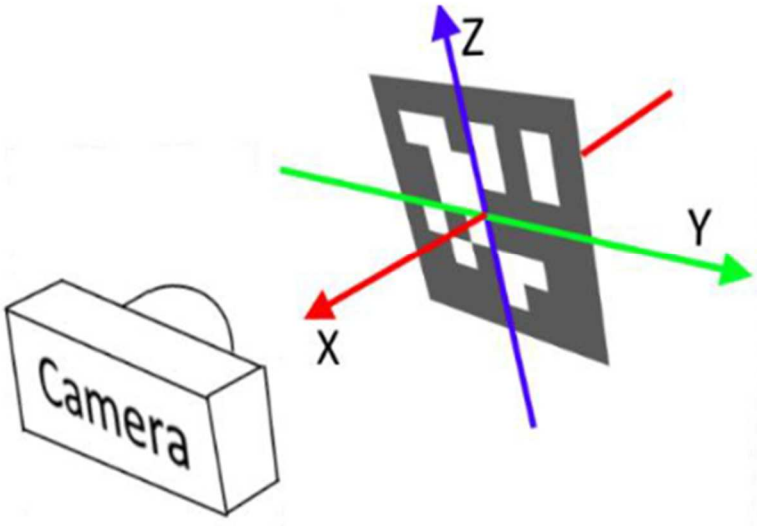


Рисунок 25 – Оси поворота маркера.

Камера находится на расстоянии 3м, испытание проводится в симуляции. По результатам испытания было выяснено, что максимальный угол тангажа и оказался 88°, а угол рысканья 72°.

### 4.5. Вывод по четвертой главе

В рамках четвёртой главы были проведены испытания разработанного программного модуля визуального обнаружения метки в симуляционной среде Unity. Испытания подтвердили работоспособность модуля: успешно выполнено обнаружение ArUco-маркера, определена максимальная дистанция детектирования – 15 метров при разрешении 640×320 пикселей. Средняя абсолютная погрешность составила 0,043 м, а средняя относительная погрешность – 0,736%, что соответствует требованиям технического задания. Также определены максимальные углы обнаружения маркера: тангаж – 88°, рыскание – 72°. Результаты испытаний демонстрируют высокую точность и эффективность программного модуля для задач автоматизированной выгрузки в мобильных робототехнических системах.

# 5. ЗАКЛЮЧЕНИЕ

В рамках выполнения выпускной квалификационной работы был разработан программный модуль для визуального обнаружения маркера, расположенного на объекте, предназначенный для интеграции в систему мобильного робота-погрузчика. Данный модуль представляет собой компонент автоматизированной системы, в которой основную роль играют визуальные сенсоры (камеры), обеспечивающие точное и быстрое обнаружение объектов в реальном времени. Проведённый анализ показал высокую актуальность и растущую потребность в автоматизации процессов на базе специализированной техники, с использованием визуальных технологий для повышения эффективности работы и безопасности. В ходе исследования также были рассмотрены различные типы маркеров, используемых для задач позиционирования и распознавания, среди которых был выбран маркер, обладающий высокой степенью надёжности и точности обнаружения в различных условиях, включая изменения углов обзора и освещения.

На этапе проектирования модуля были сформулированы основные требования, предъявляемые к его функциональности и интеграции в состав роботизированной системы. Для реализации работы с маркерами ArUco была выбрана библиотека OpenCV, как эффективный инструмент для обработки изображений и компьютерного зрения. Для обеспечения взаимодействия между различными компонентами системы, включая камеру, использовался фреймворк ROS2, который обеспечивает эффективную и надёжную передачу данных между узлами. Для реализации элемента поведения робота, связанного с принятием решений на основе полученной визуальной информации, был выбран стек NAV2, который предоставляет удобные средства для разработки интеллектуальных алгоритмов управления на основе дерева поведения (BehaviorTree).

В процессе разработки основной модуль был разделён на несколько подмодулей, каждый из которых выполняло свою задачу, связанную с обнаружением маркера, его локализацией и взаимодействием с другими системами робота. Далее были проведены серию испытаний в симуляционной среде Unity, которая позволила оценить работоспособность модуля в различных условиях, таких как изменение углов обзора камеры, дистанции до маркера и его ориентации. Результаты испытаний показали высокую точность обнаружения маркера, что свидетельствует о хорошей устойчивости системы к различным вариациям условий.

Результаты проведённых испытаний подтверждают, что разработанный программный модуль может быть эффективно использован в различных мобильных платформах и автоматизированных системах, таких как автономные погрузчики, складская техника и другие устройства, где требуется точное и надёжное визуальное позиционирование объектов. Это открывает возможности для его применения не только в промышленности, но и в других сферах, где важна высокая точность и автономность работы. Кроме того, работа может послужить основой для дальнейших исследований и разработок, направленных на расширение функциональных возможностей модуля, в том числе с использованием современных технологий, таких как нейросетевые методы и машинное обучение. Внедрение этих технологий позволит значительно повысить устойчивость системы к внешним помехам, а также улучшить её работоспособность в сложных и изменяющихся условиях окружающей среды, таких как шум, вариации освещенности или препятствия.

# БИБЛИОГРАФИЧЕСКИЙ СПИСОК


1. Мещеряков, Р.В. Обзор соревновательной робототехники. Состязания спортивных роботов / Р.В. Мещеряков, Б.В. Илюхин // Робототехника и техническая кибернетика. – 2022. – Т. 10, № 4. – С. 255-260.
2. International Federation of Robotics. –https://ifr.org/.
3. Сергиевич, Т.В. Анализ мирового рынка промышленных роботов / Т.В. Сергиевич // Вестник Гродненского государственного университета имени Янки Купалы. Серия 5. Экономика. Социология. Биология. – 2023. – Т. 13, № 1. – С. 6-14.
4. Татур, М.М. Методика модельно-ориентированного проектирования алгоритмов управления мобильными роботами / М.М. Татур, Н.С. Игнатюк, А.Д. Коников // Доклады Белорусского государственного университета информатики и радиоэлектроники. – 2024. – Т. 22, № 1. – С. 91-99.
5. Либерман, Я.Л. Принципы проектирования мобильных роботов / Я.Л. Либерман, Л.Н. Горбунова // Мехатроника, автоматика и робототехника. – 2023. – № 11. – С. 22-26.
6. Cat Machines Allow for Semi-Autonomous Control. – https://www.precisionfarmingdealer.com/articles/6244-cat-machines-allow-for-semi-autonomous-control.
7. Caterpillar. Caterpillar Inc. – https://www.cat.com/en_US/products/new/technology/command.html.
8. Ничего себе: тест-драйв беспилотного погрузчика БЕЛАЗ – управление на дистанции 2500 км. – https://abw.by/news/commercial/2018/07/11/nichego-sebe-testdraiv-bespilotnym-pogruzchikom-belaz-upravlyali-na-distancii-2500-kilometrov.
9. Volvo Construction Equipment. LX03. – https://www.volvoce.com/global/en/about-us/what-we-believe-in/innovation-at-our-core/our-innovation-concepts/lx03/.

10. First autonomous electric loaders in North America get to work. – https://electrek.co/2024/12/28/first-autonomous-electric-loaders-in-north-america-get-to-work/.
11. Rodehorst, V. Comparison and evaluation of feature point detectors / V. Rodehorst, A. Koschan //5th International Symposium Turkish-German Joint Geodetic Days. – 2006.
12. Garrido-Jurado, S. et al. Automatic generation and detection of highly reliable fiducial markers under occlusion / S. Garrido-Jurado //Pattern Recognition. – 2014. – Т. 47. – №. 6. – С. 2280-2292.
13. QR-навигация внутри помещения | Indoor-навигация с помощью QR-кодов. – https://nvgn.ru/blog/indoor-navigatsiya-qr-kodi/.
14. Kim, J. Visual Fiducial Markers for Warehouse Navigation. / J. Kim, D. Lee, Oh H. //IEEE Robotics and Automation Letters.
15. Васильев, С.В Использование лидаров в задачах построения 3D-карт и навигации мобильных роботов / С.В Васильев, А.Е. Седунов // Известия БНТУ. – 2021. – №2.
16. Закиев И.А. Алгоритм SLAM: обзор существующих методов / И.А. Закиев, Н.Ф. Нуртдинов, Е.М. Исаев [и др.] // Информационные технологии. Автоматизация. Актуализация и решение проблем подготовки высококвалифицированных кадров (ИТАП-2018) : Сборник материалов VIII международной научно-практической конференции, Набережные Челны, 19 октября 2018 года / Набережночелнинский институт (филиал) федерального государственного автономного образовательного учреждения высшего образования "Казанский (Приволжский) федеральный университет", Информационно-образовательный центр подготовки специалиста машиностроительного профиля. – Набережные Челны: Издательство Набережночелнинского института (филиала) ФГАОУ ВО «Казанский (Приволжский) федеральный университет», 2018. – С. 149-156.

17. Wu, K. Split covariance intersection filter for mobile robot localization / K. Wu et al. // IEEE Transactions on Aerospace and Electronic Systems. – 2013.
18. Fiala, M. ARTag, a fiducial marker system using digital techniques / M. Fiala //2005 IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR'05). – IEEE, 2005. – Т. 2. – С. 590-596.
19. Закиев А.А. Пилотные виртуальные эксперименты по сравнению систем координатных меток AruCo и AprilTag на устойчивость к вращению / А.А. Закиев, К.С. Шабалина, Т.Г. Цой, Е.А. Магид // Пятый Всероссийский научно-практический семинар "Беспилотные транспортные средства с элементами искусственного интеллекта" (БТС-ИИ-2019): Труды семинара, СПб, 22–24 мая 2019 года. – СПб: Общероссийская общественная организация «Российская ассоциация искусственного интеллекта», 2019. – С. 210-220.
20. Документация ROS 2: Humble Hawksbill. - https://docs.ros.org/en/humble/index.html.
21. Herath, D. The Robot Operating System (ROS1 & 2): Programming paradigms and deployment. – https://www.researchgate.net/publication/363842243_The_Robot_Operating_System_ROS1_2_Programming_Paradigms_and_Deployment.
22. NAV2. – https://nav2.org.
23. NAV2. Documentation. – https://docs.nav2.org/index.html.
24. Что такое деревья поведения и как они используются. – https://habr.com/ru/companies/mws/articles/306214.
25. Colledanchise M., Ögren P. Behavior trees in robotics and AI: An introduction. – CRC Press, 2018.
26. Что такое калибровка камеры? – https://docs.exponenta.ru/R2021a_nmtnew/vision/ug/camera-calibration.html.

27. Корк, П. Машинное зрение. Основы и алгоритмы с примерами на Matlab : руководство / П. Корк ; перевод с английского В.С. Яценкова. – М : ДМК Пресс, 2023. – 584 с.
28. Сырямкин, В.И. Информационные устройства и системы в робототехнике и мехатронике: учебное пособие для вузов / В.И. Сырямкин. – 4-е изд., стер. – СПб: Лань, 2024. – 532 с.
29. Никитин, Н.Н. Курс теоретической механики : учебник / Н.Н. Никитин. – 8-е изд., стер. – СПб: Лань, 2022. – 720 с.
30. REP-105 Coordinate Frames For Mobile Platforms. – https://www.ros.org/reps/rep-0105.html.
31. Writing a New Behavior Tree Plugin. – https://docs.nav2.org/plugin_tutorials/docs/writing_new_bt_plugin.html.
32. Платформа Unity для разработки в реальном времени. – https://unity.com/ru.