Министерство просвещения РФ
федеральное государственное бюджетное образовательное
учреждение высшего образования
«Уральский государственный педагогический университет»
Институт математики, физики, информатики
Кафедра информатики, информационных технологий
и методики обучения информатике

Система управления мобильным роботом на основе технического зрения

Выпускная квалификационная работа

Допущено к защите
Зав. кафедрой Сардак Л.В.
«__» мая 2023 г. _____

Исполнитель: Горбунов Никита Денисович
обучающийся группы ИСиТ-1931
_____

Руководитель: Шимов И.В.
старший преподаватель кафедры
ИИТиМОИ

_____

Екатеринбург – 2023

Содержание

Введение.....3

1. Теоретико-аналитическая часть.....5

1.1. Обоснование постановки задачи.....5

1.2. Выбор метода разработки.....10

1.3. Формализированное описание технического задания.....15

2. Практическая часть.....19

2.1. Модельные представления объекта разработки.....19

2.2. Описание продукта.....26

2.3. Результат апробации, техническая документация.....32

Заключение.....34

Список информационных источников.....35

Приложения.....37

Введение

В современном мире развитие отросли робототехники и автоматизации процессов является одним из перспективных направлений развития в различных сферах, начиная промышленностью и медициной, заканчивая бытовыми потребностями. Одним из ключевых направлений развития систем управления является возможность анализировать окружающую среду и принимать решения на основе полученной и обработанной информации.

Достаточно доступным способом получения большого количества информации об окружающей среде является применение технического зрения. Данный способ основан на анализе и обработке изображений, что позволяет «видеть» и распознавать объекты, измерять расстояния, определять характеристики объектов и анализировать изменение среды.

Система управления мобильным роботом на основе технического зрения представляет собой комплекс, включающий в себя аппаратную часть и программное обеспечение для обработки изображений и принятия решений на основе полученных данных.

Однако, разработка таких систем управления имеет ряд трудностей, связанных со сложностью обработки больших объемов данных, точность калибровки и качество оборудования, а также досочное освещение для корректной работы.

Реализация системы управления роботом, которая может реагировать на окружающую среду, позволяет автоматизировать процессы. Например, оператора робота можно заменить линией движения и опознавательными маркерами для выполнения разовых операций – запуска подпрограмм. Такой оптимизации человеческих ресурсов способствует автоматизация процессов.

Тысячи роботов живут своей жизнь упаковывая продукты, перемещая заказы на отправку и принимая новый товар. На момент 2015 г. на складах компании Amazon работало около 30 тысяч роботов, которые не устают и

работают круглосуточно. Развитая инфраструктура и огромный масштаб позволяют компании держать минимальные цены.

Глобальная цифровизация и конкуренция способствуют развитию автоматизации. По данным исследования, проведенного Markets and Markets research в 2017 году, ожидается, что рынок складской робототехники будет расти на 12% ежегодно, и уже к 2022 году достигло объема $4,44 миллиарда. Такой стремительный рост обусловлен переходом крупных компаний, в первую очередь — онлайн-ритейла, на роботизированную организацию складов. Россия тоже не остается в стороне, но идея автоматизации в этой отрасли только набирает свои обороты.

Чтобы изучить тему технического зрения для управления мобильным роботом были поставлены следующая цель и задачи:

Цель: разработать систему управления мобильным роботом на основе технического зрения.

Задачи:

- проанализировать рынок существующих решений;
- изучить принцип работы машинного зрения;
- изучить возможности мобильного робота RoboMaster S1;
- разработать систему в соответствии с техническим заданием;
- подготовить техническую и сопроводительную документацию.

Продукт разработки: система управления мобильным роботом RoboMaster S1.

1. Теоретико-аналитическая часть

1.1. Обоснование постановки задачи

Одной из существующих задач робототехники является навигация в пространстве. Данная задача актуальна в случаях использования мобильного робота в автономном режиме. Для решения данной задачи робот должен уметь определять свое местоположение в пространстве, выполнять анализ окружающей среды на наличие препятствий для перемещения и выбор маршрута движения без участия человека.

Для успешного определения своего местоположения в пространстве бортовая система робота должна уметь строить маршрут передвижения, управлять параметрами поворота колес и их скорости вращения, правильно интерпретировать сведения об окружающем мире, получаемые от датчиков, а так же постоянно отслеживать собственные координаты. Данная задача интересна не только робототехникам, но и специалистам из других областей – космической, авиационной и автомобильной.

Над вопросом определения своего местоположения в пространстве люди задумывались с самых давних времен. Мореплаватели и путешественники использовали карты, компасы, навигационные отметки, небесные объекты и т.д. В итоге, были выделены следующие схемы навигации:

- глобальная – определение абсолютных координат устройства при движении по длинным маршрутам;
- локальная – определение координат устройства по отношению к некоторой точке;
- персональная – позиционирование частей устройства в пространстве и взаимодействие с объектами, что востребовано для устройств с наличием манипулятора.

Существует еще одна классификация по отношению к видам получения информации:

- пассивная – прием информации о собственных координатах и других характеристик своего движения от внешних источников;
- активная – расчет информации о собственных координатах посредством собственных устройств.

Одним из самых популярных, или же знакомых почти каждому человеку способов позиционирования, является GPS (*англ. Global Positioning System* – Глобальная Система Позиционирования) – спутниковая система навигации, обеспечивающая измерение расстояния и определение координат.

Точность определения положения зависит от большого числа негативных факторов, начиная астрономическими явлениям и заканчивая неудачной конфигурацией спутников во время измерений. Тем не менее, такая погрешность редко выходит за рамки 10–15 метров для полно охватной американской системе NAVSTAR. При этом, данный «точный» канал для широкого пользования остается под контролем правительства США, которое может в любой момент закрыть его для пользования, что снизит точность до сотни метров погрешности. Конечно, в мире есть аналогичные действующие разработки от других компаний и специализированных целей. [6]

Не стоит забывать про закон робототехники о точности навигации – ошибка в определении собственных координат не может превышать размера автономного аппарата. Нарушение данного требования может привести к столкновению с другими объектами такого же или меньшего размера. Поэтому данный способ ориентации больше подходит для крупногабаритных и местами предсказуемых объектов: рейсовые самолеты, корабли и т. д.

Кроме того, применение данного способа ориентации в пространстве становится проблематичным, когда нужно использовать мобильного робота в пространстве, так как GPS-навигация чувствительна к физическим помехам: рельеф местности и потолочные перекрытия. Таким образом, данная система не

подходит для использования, как основная в задачах навигации и ориентации в пространстве мобильных роботов.

Другая пассивная концепция, но уже локальной навигации подразумевает использование радиомаяков заключается в размещении в зоне действий робота источников радиосигналов, которые обрабатываются бортовым микропроцессором. Но так как радиомаяки располагаются в фиксированных точках некоторого маршрута, аппарат теряет возможность обходить препятствия или выбирать альтернативный путь движения. [3]

Схожим образом устроены системы с инфракрасными каналами. Радиосигналы не подходят для ориентирования в зоне прямой видимости из-за огибания препятствий, сложности настройки и проблем помехозащищенности. Видимый диапазон неудобен в силу естественных причин. С другой стороны ИК-диапазон удобен в зоне прямой видимости, он позволяет четко ориентироваться в пространстве, обладает неплохой помехозащищенностью. Однако, из-за схожести в принципе работы с радио маяками, данная система не подходит для использования в условиях с возможными препятствиями.

Успешно применяются и активные навигационные схемы, такие как инерционные навигационные системы (ИНС). Инерциальная навигация – это метод навигации и управления движением, основанный на свойствах инерции, являющийся автономным. Данная система позволяет отслеживать мобильные устройства автономно. Основная ее идея заключается в том, чтобы, основываясь на некоторой начальной информации о местонахождении объекта, отслеживать его перемещение на основании показаний датчиков мобильного устройства – акселерометра и гироскопа.

Сущность инерциальной навигации состоит в определении ускорения объекта и его угловых скоростей с помощью установленных на движущемся объекте приборов и устройств, а по этим данным – местоположения (координат) этого объекта, его курса, скорости, пройденного пути и др., а также

в определении параметров, необходимых для стабилизации объекта и автоматического управления его движением. Это осуществляется с помощью:

1. Датчиков линейного ускорения (акселерометров).
2. Гироскопических устройств, воспроизводящих на объекте систему отсчета и позволяющих определять углы поворота и наклона объекта, используемые для его стабилизации и управления движением.
3. Вычислительных устройств (ЭВМ), которые по ускорениям (путем их интегрирования) находят скорость объекта, его координаты и др. параметры движения.

Преимущества методов инерциальной навигации состоят в автономности, помехозащищенности и возможности полной автоматизации всех процессов навигации. Благодаря этому методы инерциальной навигации получают все более широкое применение при решении проблем навигации надводных, подводных и воздушных судов, космических судов и аппаратов и других движущихся объектов.

Главный недостаток механических ИНС — накопление ошибок измерения за время активной работы. Кроме того, ИНС малоэффективны в случаях, когда скорость объекта часто и резко меняется. Они также плохо подходят для задач навигации роботов среднего и малого размеров. Навигационные версии гироскопов должны устанавливаться на стабильной платформе, к тому же цена их высока.

Простейший вариант хорошо известного всем автолюбителям активного навигационного устройства — одометр. Он периодически измеряет скорость вращения колеса и, так как диаметр последнего известен, определяет пройденный путь. Но колеса любого двигающегося устройства не идеально ровные, из-за чего реальная длина покрышки всегда будет отличаться от рассчитанной, к тому же они могут прокручиваться вхолостую или проскальзывать, а сам одометр под воздействием внешних и внутренних помех постоянно накапливает ошибки измерения. Однако несмотря на все эти

недостатки данная технология применяется очень активно. Иногда, одометры — это единственно возможное навигационное решение (например, при нахождении робота на другой планете).[8]

Первые модели промышленных роботов с более или менее автономной навигацией, передвигались по маршруту, жестко заданному с помощью электрических кабелей, проложенных под полом по маршруту. На роботах устанавливались несложные устройства приема электромагнитного излучения кабеля, позволявшие определять направление перемещения. Аппараты могли двигаться по различным маршрутам благодаря тому, что по нескольким кабелям передавался сигнал с разной частотой. Но такая схема была дорогой и негибкой.

С появлением первых систем машинного зрения удалось отказаться прокладывания кабелей и перейти к навигации по ярко нарисованным линиям на полу. Робот с помощью камеры следил за такой линией и самостоятельно двигался вдоль нее. Простые реализации данной системы часто сталкивались с различными проблемами, которые критично влияли на работоспособность: износ линий, перекрытие области распознавания предметами, другими аппаратами или людьми, а на перекрестках маршрутов, роботы обычно теряются или останавливаются, что требует вмешательства человека [4].

Чтобы определять наличие препятствий в системе локальной навигации используют генераторы ультразвуковых и инфракрасных сигналов, а также лазерных дальномеров. Однако, эффективность и точность подобных устройств в большей степени зависит от характеристик среды. К тому же, данными датчиками мы контролируем только расстояние до какого-либо объекта, определить характер и параметры объекта не предоставляется возможным. Поэтому данные устройства являются больше вспомогательными для получения более точной информации о внешних факторах. [10]

С течением времени, система технического зрения научилась распознавать препятствия с помощью более совершенных алгоритмов

обработки изображений.[13] В качестве основных методов решения задачи определения препятствий можно выделить следующие: метод стереовидения, [11] метод триангуляции,[5] метод определения расстояния до препятствия с использованием динамически меняющегося изображения.[9]

Нельзя не упомянуть гибридные бортовые системы, которые используют навигационные средства всех видов, но занимается прежде всего оценкой окружающей обстановки, анализом выполняемого задания и принятием решений. Аппараты с данным типом системы пытаются построить образ окружающей среды, для анализа пространства и дальнейших действий, после чего формируется маршрут и выполняется движение по нему, постоянно сопоставляя со сгенерированной картой пространства. Иногда, перед роботом, не стоит задача построения карты помещения, так как он высчитывается другими способами и загружается в саму систему, но тогда опять появляется задача сопоставления реальных координат робота с загруженной в память картой местности.

Таким образом, для решения большинства задач, стоящих перед мобильным роботом, системы технического зрения будет достаточно, чтобы разработать систему управления, которая будет выполнять ряд определенных задач без вмешательства человека.

1.2. Выбор метода разработки

Разработка системы управления мобильным роботом на основе технического зрения – это сложный комплекс, использующий видеоизображение с камер или данные с сенсоров, чтобы управлять роботом на основе анализа окружающей среды.

Основные компоненты системы могут включать в себя:

1. Устройство видеозахвата, которое позволяет собирать изображения окружающей среды.

2. Алгоритмы обработки изображений, которые используются для распознавания объектов и других характеристик окружающего пространства.
3. Алгоритмы управления роботом на основе анализа данных, полученных от камеры.
4. Двигатели для осуществления движения и возможности взаимодействия с окружающими объектами.
5. Компьютер, контроллер или другое устройство управления для контролирования работы системы управления.

Для начала работы следует определиться с аппаратной частью. Реализация программного обеспечения принципиально не зависит от выбора аппаратного комплекса, так как это математические алгоритмы, описанные формулами. Общая технология работы состоит в получении двумерного изображения пространства через устройство видеозахвата, обработка изображения и анализ, отправка сигналов моторам. Однако, сложность реализации этой программы на прямую зависит от технических характеристик аппаратного комплекса и предоставленных возможностей по разработке.

Проанализировав рынок, можно выделить несколько групп технических средств для разработки мобильных роботов. Некоторые из них:

1. Микроконтроллеры: Arduino, Raspberry Pi и другие, которые используются для управления двигателями, датчиками, аналоговыми и цифровыми устройствами.
2. Датчики: ультразвуковые, инфракрасные, гироскопы, акселерометры, компасы, камеры и многие другие для получения информации о окружении.
3. Двигатели и сервоприводы: DC-моторы, шаговые моторы, сервоприводы и другие, которые обеспечивают передвижение мобильного робота.
4. Колеса и шасси используются для передвижения мобильного робота.
5. Механические компоненты: крепежи, шарниры, кабельные каналы и другие для конструирования корпуса мобильного робота.

6. Беспроводные модули связи: Bluetooth, Wi-Fi, ZigBee, LoRa и другие, которые обеспечивают беспроводное соединение между мобильным роботом и управляющим устройством.
7. Программное обеспечение: ROS (Robot Operating System), Arduino IDE, Python, C++, MATLAB и другие для программирования мобильных роботов и управления ими.

В зависимости от задач, которые должен выполнять мобильный робот, могут использоваться различные технические средства и комбинации из них. Если не стоит задача самостоятельно разработать и собрать мобильного робота, то можно воспользоваться готовыми сборными комплектами или устройствами.

На рынке образовательных конструкторов существует множество учебных наборов, которые позволяют создавать собственные системы управления на основе технического зрения. Некоторые из них:

1. LEGO Mindstorms – это конструктор, который позволяет создавать программируемых роботов. Он содержит в себе датчики и камеры, которые могут быть использованы для разработки систем управления на основе технического зрения.
2. VEX Robotics – это платформа для создания роботов, которая содержит в себе мощный микроконтроллер и множество датчиков, включая камеры. Она может быть использована для разработки систем управления на основе технического зрения.
3. Raspberry Pi – это небольшой одноплатный компьютер, который можно использовать для управления роботами на основе технического зрения. Он содержит в себе мощный процессор и множество интерфейсов, которые позволяют подключать различные датчики и камеры.
4. Arduino – это микроконтроллер, который может быть использован для управления роботами на основе технического зрения. Он содержит в себе множество интерфейсов, которые позволяют подключать различные датчики и камеры.

5. DJI RoboMaster – это программируемый робот, который был создан для соревнований в боевых играх. Он содержит в себе камеры и другие датчики, которые позволяют разработать систему управления на основе технического зрения.

Это лишь некоторые примеры конструкторов и роботов, которые могут быть использованы для разработки систем управления на основе технического зрения.

Проведем анализ данных технических решений.

LEGO Mindstorms и другие производные продукты, вдохновленные успехом данной компании, пользуются большим спросом в сфере образования. Простота в сборке деталей, подключении периферийных устройств (моторы, датчики и контроллеры) и наличие блочной среды программирования позволяют создавать различных роботов детьми младшего возраста. Однако, данный тип программирования и возможности управляемого блока накладывают ряд ограничений в реализации сложных программ и подключении сторонних модулей. Также, стоит отметить, что датчики и моторы имеют существенную неточность в работе, которая влияет на эффективность работа.

VEX Robotics – еще один представитель конструкторов для создания роботов. Данная платформа имеет контроллер с расширенными возможностями подключения сторонних компонентов, достаточно качественные моторы, более гибкую возможность программирования, в некоторых комплектах детали для сборки не пластиковые, а металлические. Для кого-то плюсы данного набора окажутся минусами, так как порог входа выше и от пользователя требуется знать основы электроники, механики и программирования.

Иногда встречаются готовые комплекты на базе Arduino. Тогда пользователю доступен большой пласт для экспериментов со сторонней электроникой, а также отсутствует проблема с поиском каких-нибудь деталей для замены. Данный набор наследует минусы VEX Robotic с высоким порогом входа, а также невозможность использования данной платформы для системы

технического зрения с видеокамерой в самостоятельном режиме. В дополнение к Arduino требуется модуль для анализа изображения: компьютер или Raspberry Pi с постоянным соединением, в связи с малой вычислительной мощностью. Тогда одно устройство получает видеосигнал, обрабатывает его и отправляет команды на микроконтроллер. Такое взаимодействие вносит ряд сложностей с подключением устройств.

А вот Raspberry Pi может самостоятельно работать с видеокамерой и управлять электроникой. Наличие различных интерфейсов позволяют подключать не только классические электронные компоненты, но и специализированные, иногда выпущенные целенаправленно для использования с данным одноплатным компьютером. Программировать данное устройство можно на любых языках, в том числе и Python, что дает полный доступ к библиотекам для анализа изображений с камеры. Минусы такие же, как и у предшественников с добавлением сложности в прошивке и настройки системы. Следует отметить, что не все компоненты электроники могут работать исправно и ошибка в сборке электрической цепи может вызвать поломку не только компонентов, но управляющей платы.

В наборе RoboMaster S1 использование технического зрения с помощью видеокамеры изначально создавалось, как ключевая возможность, поэтому подпрограммы для распознавания людей, жестов, меток и линий доступны из коробки. Отметим, что у данного набора высокое качество исполнения деталей, хорошая точность датчиков и моторов, а итоговая собранная модель на железном каркасе является крепкой, по сравнению с LEGO и VEX. Роботом управляет микроконтроллер, но для программирования требуется подключение к внешнему устройству. На персональный компьютер или смартфон устанавливается фирменное приложение со встроенной средой программирования на языке Python. На самом деле, существует готовая библиотека для программирования, а данное приложение является интерфейсом взаимодействия с библиотекой, что позволяет создавать

программы не только строками кода, но и блочным программированием. Также есть возможность установить данную библиотеку отдельно от всего приложения и реализовать программу с использованием сторонних библиотек: OpenCV, TanserFlow или самостоятельно углубиться в обработку данных и искусственный интеллект. Данный функционал реализуем из-за особенности взаимодействия робота с устройством: вычисления проводятся на компьютере, а результаты обработки, в виде команд, отправляются на робота по Wi-Fi соединению. Отличие такого взаимодействие от Arduino + RasberiPy/ПК заключается в том, что имеется официальное приложение для единственного способа подключения – Wi-Fi соединение, дальнейшая синхронизация двух устройств уже заложена в эту программу, после подключения можно сразу писать программу, состоящую из команд для робота. Минусами данного комплекта являются: высокая стоимость, программирование на Python требует определённых знаний языка и документации DJI Robomaster, ограниченность в выборе компонентов и дополнительных модулей, ограниченная гибкость конструкции для каких-либо изменений.

Исходя из поставленной цели: разработать систему управления мобильным роботом на основе технического зрения, платформа DJI Robomaster S1 подходит больше всего для дальнейшей реализации. Данная платформа имеет все основные компоненты системы, удобную и понятную среду разработки со встроенным функционалом для работы с техническом зрением.

1.3. Формализированное описание технического задания

Техническое задание на разработку информационной системы составлен на основе ГОСТ 34.602-89 «Техническое задание на создание автоматизированной системы».

1. Общие сведения.

1.1. Название продукта разработки (проектирования).

«Система управления мобильным роботом на основе технического зрения».

1.2. Название организации-заказчика.

ФГБОУ ВО «УрГПУ».

1.3. Плановые сроки начала и окончания работ.

В соответствии с планом выполнения ВКР.

2. Назначение и цели создания системы.

2.1. Назначение системы.

Автоматизация процесса сортировки и транспортировки объектов на базе машинного зрения конструктора DJI Robomaster S1.

2.2. Цели создания системы.

Система создается с целью отладки алгоритма движения по траектории и анализа меток для получения информации о транспортируемых объектах.

В результате создания системы должны быть продемонстрированы следующие возможности:

1. Сортировка объектов на основе анализа меток.
2. Движение по цветным траекториям.
3. Автономная работа системы.

3. Требования к системе.

3.1. Требования к системе в целом.

- система должна работать в автономном режиме после запуска;
- соблюдение всех требований для успешной работы модулей искусственного интеллекта согласно руководству пользователя v1.8;
- траектории движения должны быть синего, зеленого или красного цвета;
- радиус поворота траектории должен соответствовать требованиям документ;
- использовать опознавательные маркеты, которые входят официальный список;

- опознавательные маркеры должны быть в виде белых символов на красном фоне;
- красная зона опознавательных маркеров не должна быть заблокирована посторонними объектами;
- система реализуется с помощью языков программирования: scratch, встроенный в среду разработки программы «RoboMaster», Python, встроенный в среду разработки программы «RoboMaster», Python версии 3.8 и RoboMaster SDK 0.1.1.68 для программирования в других средах разработки.
- неизменность планировки складских помещений после запуска системы.

3.2. Аппаратные требования.

Аппаратные требования устройства, используемого системой согласно DJI RoboMaster S1 руководству пользователя v1.8.

3.3. Указание системного программного обеспечения.

Официальное внутренне ПО набора RoboMaster S1.

3.4. Указание программного обеспечения, используемого для реализации.

Официальное приложение RoboMaster для Windows 7 64bit и выше.

3.5. Форматы входных данных.

В качестве входных данных используется видео – поток для дальнейшей обработки алгоритмами ИИ.

3.6. Источники данных.

В качестве источников данных используются опознавательные маркеры и линии траектории движения.

4. Требования к документированию.

4.1. Перечень сопроводительной документации.

RoboMaster S1 руководство пользователя v1.8.

4.2. Требования к содержанию отдельных документов.

В руководстве пользователя содержится информация про схему S1, подготовку к использованию, используемые модули и функции, технические

характеристики устройства, процесс обновления ПО, настройка портов ШИМ, калибровка S1.

5. Порядок сдачи-приема продукта.

5.1. В соответствии с планом выполнения ВКР.

2. Практическая часть

2.1. Модельные представления объекта разработки

Перед началом разработки системы следует ознакомиться с аппаратной конфигурации робота DJI Robomaster S1. [1Error: Reference source not found]

Таблица 1.

Камера

Параметр	**Значение**
Угол обзора	120°
Макс. разрешение фото	2560×1440
Макс. разрешение видео	FHD: 1080/30 кадров/с HD: 720/30 кадров/с
Макс. битрейт видео	16 Мбит/с
Фотоформаты	JPEG
Видеоформаты	MP4
Матрица	CMOS 1/4 дюйма, число эффективных пикселей: 5 млн
Диапазон рабочих температур	-10...+40°C

Таблица 2.

S1

Параметр	**Значение**
Масса	Около 3,3 кг
Размеры	320×240×270 мм (длина × ширина × высота)
Диапозон скорости шасси	0–3,5 м/с (вперед) 0–2,5 м/с (назад) 0–2,8 м/с (в стороны)
Макс. скорость вращения шасси	600°/с

Таблица 3.

Бесщеточный двигатель M3508I

Параметр	**Значение**
Макс. скорость вращения	1000 об/мин
Макс. момент силы	0,25 Н·м
Мощность на выходе	19 Вт
Диапозон рабочих температур	-10...+40°C
Опертаор	FOC
Метод управления	Замкнутая система регулирования скорости
Защита	Защита от избыточного напряжения. Защита от перегрева. Плавный пуск. Защита от короткого замыкания. Обнаружение аномальной работы чипа и датчика

Таблица 4.

Интелектуальный контроллер

Параметр	Значение
Задержка сигнала	Подключение по Wi-Fi: 80–100 мс. Подключение по роутеру: 100–120 мс (при отсутствии препятствий и помех)
Качество трансляции	720p/30 кадров/с
Макс. битрейт трансляции	6 Мбит/с
Диапазон рабочих частот	2,4 ГГц, 5,1 ГГц
Режим работы	Соединение по Wi-Fi, соединение по роутеру
Макс. дальность передачи сигнала	Подключение по Wi-Fi: FCC: 2,4 ГГц, 140 м CE: 2,4 ГГц, 130 м SRRC: 2,4 ГГц, 130 м MIC: 2,4 ГГц, 130 м Подключение по роутеру: FCC: 2,4 ГГц, 190 м CE: 2,4 ГГц, 180 м SRRC: 2,4 ГГц, 180 м MIC: 2,4 ГГц, 180 м
Мощность передатчика (ЭИИМ)	2,4–2,4835 ГГц FCC: ≤30 дБм SRRC: ≤20 дБм CE: ≤19 дБм MIC: ≤20 дБм 5,15–5,25 ГГц FCC: ≤30 дБм SRRC: ≤23 дБм CE: ≤20 дБм MIC: ≤23 дБм
Стандарт передачи	IEEE802.11a/b/g/n

Таблица 5.

Аккаумлятор Intelligent Battery

Параметр	Значение
Емкость	2400 мАч
Номинальное напряжение зарядки	10,8 В
Максимальное напряжение зарядки	12,6 В
Тип	Литий-полимерный 3S
Энергия	25,92 Вт/ч
Масса	169 г
Диапазон рабочих температур	-10...+40°C
Диапазон температур зарядки	+5...+40°C
Максимальная мощность зарядки	29 Вт
Срок службы аккумулятора (в использовании)	35 минут (измерено при постоянной скорости 2 м/с на плоской поверхности)
Срок службы аккумулятора (в режиме ожидания)	Около 100 минут

Таблица 6.

Зарядное устройство

Параметр	Значение
Вход	100–240 В, 50/60 Гц, 1 А
Выход	Разъем: 12,6 В = 0,8 А или 12,6 В = 2,2 А
Напряжение	12,6 В
Номинальное напряжение	28 Вт

Для управления и программирования робота DJI Robomaster S1 используется приложение DJI Robomaster, которое доступно для скачивания с официального сайта для ПК на операционных системах Windows и masOS, а также на мобильные устройства на базе Android и iOS.

Таблица 7.

Системные требования для установки приложения DJI Robomaster

Операционная система	Параметр	Значение
Мобильное устройство		
Androd	Минимальная версия операционной системы	iOS 9.0
	Совместимость	iPhone, iPad и iPod touch
iOS	Минимальная версия операционной системы	Android 5.0
	Совместимость	Android-смартфоны и планшеты
Персональный компьютер		
Windows и macOS	ОС	Windows 7 или более новая версия, macOS 10.13 или более новая версия
	Процессор	Intel Core i3 или более производительный
	Оперативная память	не менее 4 ГБ
	Свободное место на жестком диске	не менее 5 ГБ
	Графический процессор	NVIDIA GeForce GTX 660 или более производительный, или AMD Radeon HD 7850 или более производительный, или Intel HD Graphics 4000 или более производительный
	Порт	USB 2.0

Для программирования робота DJI Robomaster S1 используется язык Python [7]. Python – это высокоуровневый язык программирования, который

был разработан Гвидо ван Россумом и впервые выпущен в 1991 году. Он обладает простым и понятным синтаксисом, что делает его очень доступным для новичков в программировании. Python изначально был создан как язык общего назначения, но с течением времени он стал широко применяться в различных областях, включая веб-разработку, научные исследования, анализ данных, искусственный интеллект и робототехнику.

Python предоставляет простой и понятный синтаксис, что делает его доступным даже для новичков в программировании. Он также обладает богатой экосистемой инструментов и библиотек, которые значительно упрощают разработку роботов, включая функции компьютерного зрения, машинного обучения и управления аппаратурой.

Благодаря своей популярности и обширному сообществу разработчиков, Python обеспечивает поддержку и ресурсы для программирования роботов на различных уровнях сложности. От написания простых скриптов для управления движением робота до разработки сложных алгоритмов восприятия окружающей среды и принятия решений, Python предоставляет программистам все необходимое для создания мощных и интеллектуальных роботов.

Программисты могут использовать Python для разработки различных видов роботов, включая мобильных роботов, промышленные манипуляторы, дроны и автономные транспортные средства. Python обеспечивает широкий выбор библиотек и фреймворков, специально разработанных для робототехники, таких как ROS (Robot Operating System), Pygame и OpenCV.

OpenCV (Open Source Computer Vision) – это библиотека с открытым исходным кодом, специально разработанная для обработки изображений и компьютерного зрения. Она предоставляет набор функций и алгоритмов, которые позволяют разработчикам работать с изображениями и видео, извлекать информацию из них, выполнять обнаружение объектов, распознавание лиц, трекинг движущихся объектов, анализ и синтез

изображений, а также решать другие задачи компьютерного зрения [2Error: Reference source not found].

OpenCV была разработана с целью предоставить универсальный и эффективный инструментарий для обработки изображений в различных приложениях. Библиотека написана на C++ и имеет интерфейсы для использования на различных языках программирования, включая Python, Java, C# и другие.

Основные особенности OpenCV:

1. Обработка и анализ изображений: предоставляет множество функций для чтения, записи, изменения размера, фильтрации, улучшения качества и преобразования изображений.
2. Распознавание и обнаружение объектов: включает в себя алгоритмы для распознавания и классификации объектов на изображениях. Это включает в себя обнаружение лиц, распознавание объектов на основе признаков (например, особенностей SIFT, SURF), детектирование и отслеживание движущихся объектов и другие методы машинного зрения.
3. Работа с видео: позволяет работать с видеофайлами и потоками видео. Библиотека поддерживает чтение, запись, обработку и анализ видеоданных, включая выделение движения, извлечение кадров, стабилизацию изображения и другие операции.
4. Машинное обучение: предоставляет возможности для применения методов машинного обучения в задачах компьютерного зрения. Библиотека содержит реализации алгоритмов машинного обучения, таких как метод опорных векторов (SVM), наивный Байесовский классификатор, случайный лес и другие.

OpenCV широко используется в различных областях, таких как робототехника, автоматизация, видеонаблюдение, медицина, анализ данных, дополненная реальность и многое другое.

Система технического зрения использует данные технологи для управления роботом. На рисунке (Рис. 1) изображена схема блоков обработки информации для для дальнейшего использования.

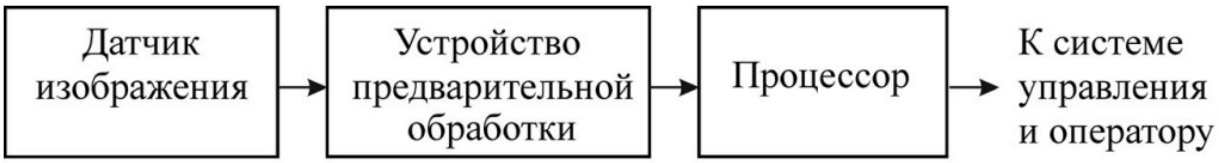

Рис. 1. Схема системы технического зрения

В компьютерах используется цифровое изображение, которое представляет собой полученную от датчиков изображения совокупность строк и столбцов пикселов (pixels — сокращение слов picture element — элемент изображения), предварительно преобразованных в цифровую форму. Каждый пиксел характеризуется интенсивностью (яркостью), которая представляется числом. Обычно используют однобайтовые (8-битовые) числа, дающие числовые значения от 0 до 255, реже — 10 битовые (1024 значения).

Каждый кадр изображения обрабатывается алгоритмом распознавания искусственного ориентира на наличие паттернов: линия или метка. Изображение преобразуется в двоичное черно-белое изображение, где линия представлена белым цветом, а фон - черным. Для этого применяются методы пороговой обработки, которые позволяют разделить линию и фон на основе яркости или цвета пикселей [12]. Выделение объектов по цвету в данном алгоритме осуществляется путём выбора фиксированных диапазонов тона, насыщенности и яркости для каждого из выделяемых цветов. Такой способ позволяет ускорить работу алгоритма, однако он чувствителен к изменению освещения наблюдаемой сцены, к бликам от ярких источников света, к изменению цветопередачи видеокамеры В дальнейшем применяются алгоритмы компьютерного зрения для обнаружения линии на бинарном изображении. Это может включать поиск границ, выделение контуров или использование специализированных алгоритмов, таких как преобразование Хафа для обнаружения прямых линий. После обнаружения линии, вычисляются

ее параметры, такие как положение (Рис. 2), угол (Рис. 3) и форма (Рис. 4). Это может быть достигнуто путем аппроксимации контура линии или анализа геометрических характеристик линии. На основе положения линии робот генерирует управляющий сигнал, который позволяет ему следовать по линии. Например, если линия отклоняется влево, робот может корректировать свое движение вправо для возвращения на линию.

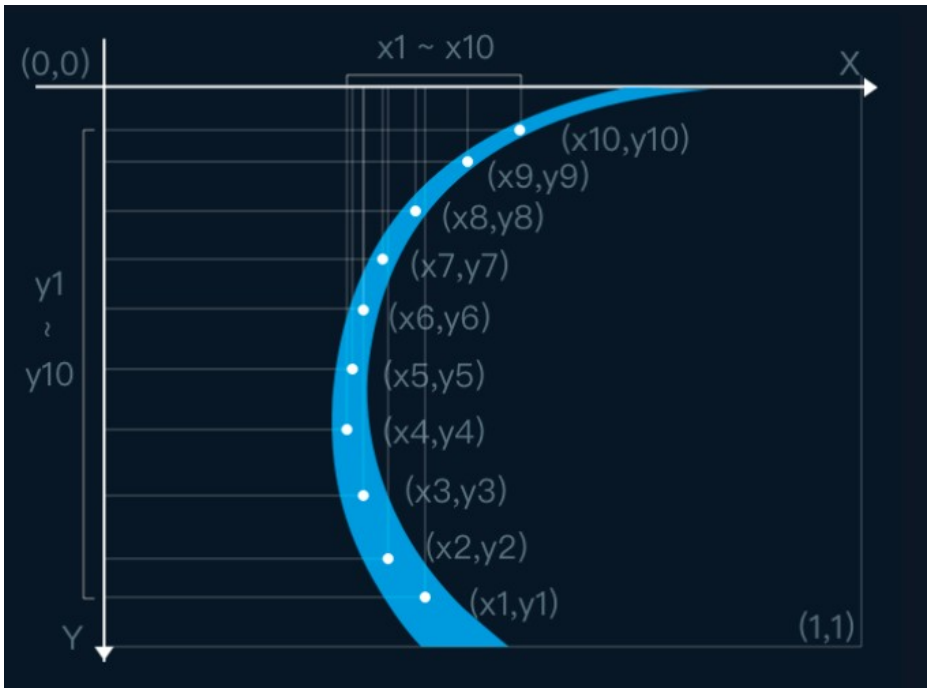

Рис. 2. Построение точек по траектории линии

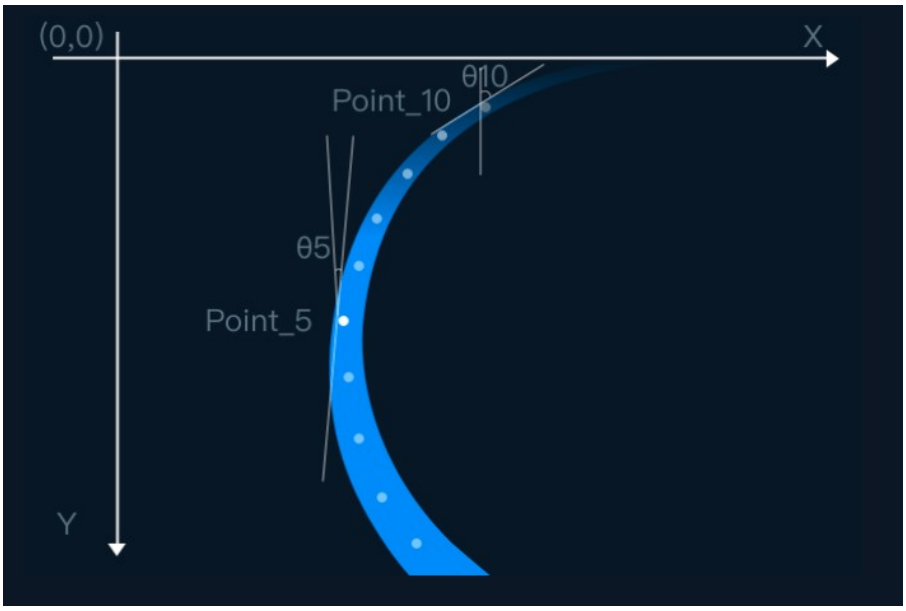

Рис. 3. Действительный тангенс угла

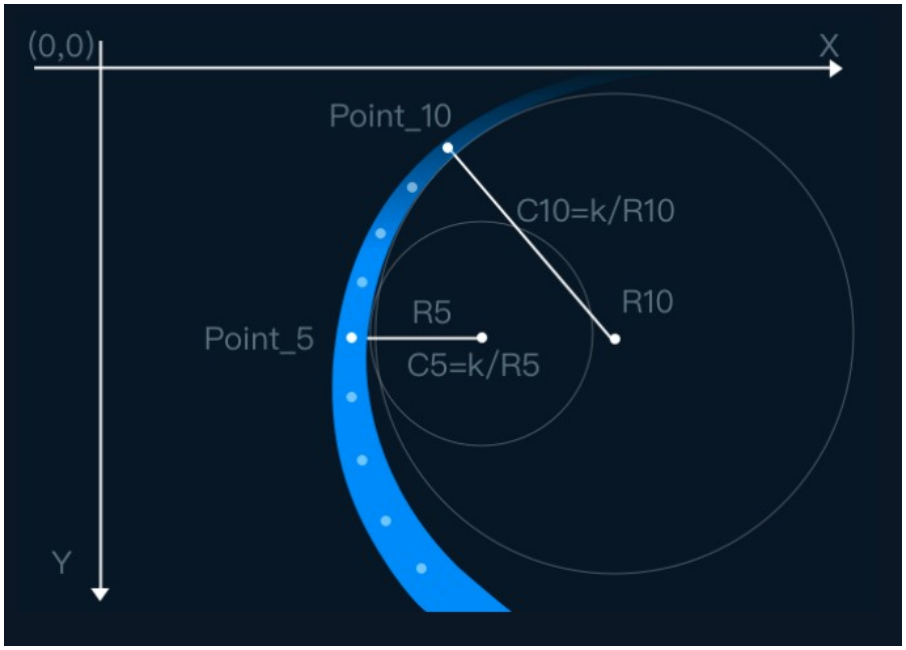

Рис. 4. Кривизна линии

Анализ траектории работает постоянно из-за внутреннего бесконечного программного рабочего цикла. Частота обработки сигнала напрямую зависит от частоты съемки камеры и скорости вычислений контроллера.

2.2. Описание продукта

Определим задачи для системы управления мобильным робот:

1. Выезд со стоянки робота до линии движения.

2. Въезд и выезд со склада погрузки.

3. Въезд и выезд со склада разгрузки.

4. Взаимодействие с объектами.

5. Анализ опознавательных маркеров.

Для начала разработаем схему трассы (Рис. 5), в соответствии с задачами и руководством пользования. На трассе присутствует зона старта и финиша (Start/Finish), точка подбора объектов (X), две линии разных цветов и два соответствующих склада (P).

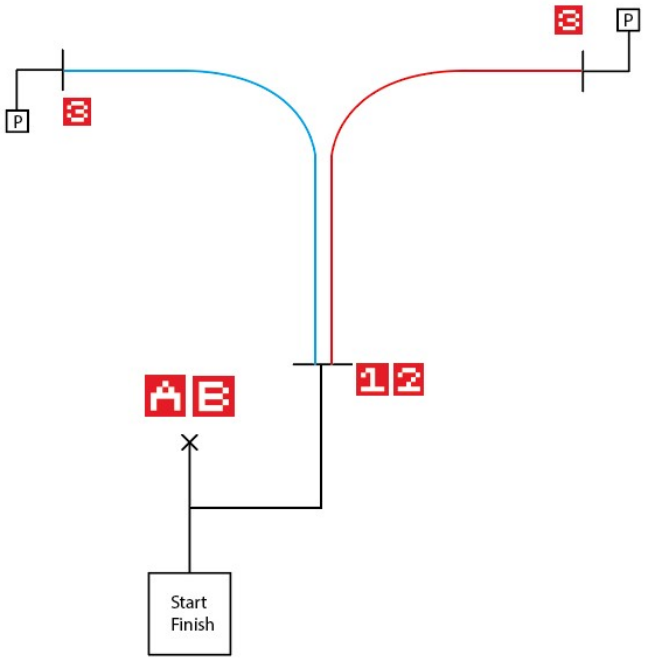

Рис. 5. Схема тестовой трассы

Составим общий алгоритм логики работы программы (Рис. 6).

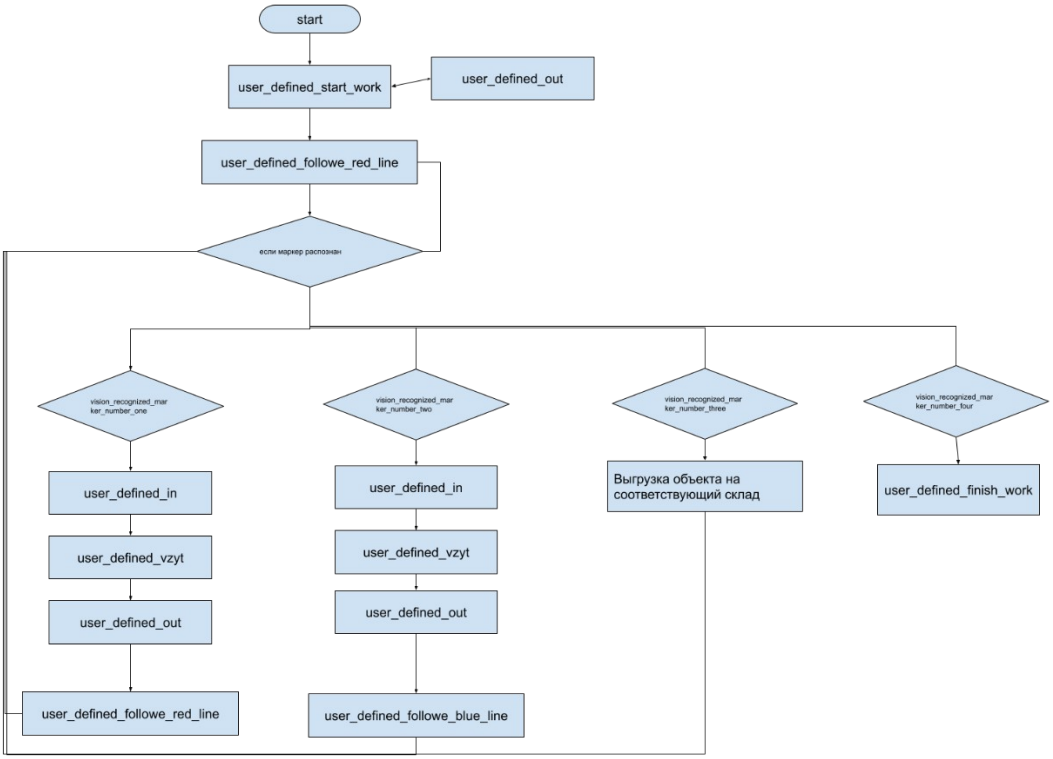

Рис. 6. Логика работы программы

Так как система управления разрабатывалась во внутренней среде разработки программы DJI RoboMaster, то у разработчика есть доступ к методам класса робота, которые уже входят в состав robomaster SDK. «chassis_ctrl» – объект шасси, с которым можно взаимодействовать, вызывая его методы. Самыми частыми методами взаимодействия с шасси являются:

- set_trans_spees() – задать скорость смещения;
- move_with_distance() – смещаться в направлении;
- rotate_with_degree() – поворот шасси.

Зная постоянное место стоянки робота и постоянное место нахождения линии движения, следует написать постоянный скрипт выезда мобильного робота с точки старта на линию движения (Рис. 7) и функцию заезда в стартовую точку (Рис. 8).

```
def user_defined_start_work():
    global variable_lastMarker
    global variable_x
    global variable_cord
    global list_lastMarker
    global list_lineList
    global pid_Follow_line
    chassis_ctrl.set_trans_speed(0.2)
    chassis_ctrl.move_with_distance(0,0.5)
    user_defined_out()
```

Рис. 7. Функция user_defined_start_wordk();

```
def user_defined_finish_work():
    global variable_lastMarker
    global variable_x
    global variable_cord
    global list_lastMarker
    global list_lineList
    global pid_Follow_line
    chassis_ctrl.stop()
    time.sleep(2)
    chassis_ctrl.set_trans_speed(0.2)
    chassis_ctrl.move_with_distance(0,0.6)
    chassis_ctrl.rotate_with_degree(rm_define.clockwise,90)
    chassis_ctrl.move_with_distance(0,0.8)
    chassis_ctrl.rotate_with_degree(rm_define.anticlockwise,90)
    chassis_ctrl.move_with_distance(0,0.5)
    chassis_ctrl.rotate_with_degree(rm_define.anticlockwise,180)
    chassis_ctrl.stop()
```

Рис. 8. Функция user_defined_finish_work();

В данную функцию входит функция user_defined_start_in(), которая служит функцией заезда и выезда на траекторию линии движения с точки старта и точки взятия объекта (Рис. 9).

```
def user_defined_in():
    global variable_lastMarker
    global variable_x
    global variable_cord
    global list_lastMarker
    global list_lineList
    global pid_Follow_line
    chassis_ctrl.stop()
    time.sleep(2)
    chassis_ctrl.set_trans_speed(0.2)
    chassis_ctrl.move_with_distance(0,0.6)
    chassis_ctrl.rotate_with_degree(rm_define.clockwise,90)
    chassis_ctrl.move_with_distance(0,0.8)
    chassis_ctrl.rotate_with_degree(rm_define.clockwise,90)
    chassis_ctrl.stop()
```

Рис. 9. Функция user_defined_in();

После выезда на траекторию движения запускается подпрограмма движения по лини (Рис. 10). Классом, отвечающим за работу распознавания объектов, является «vision_ctr». В цикле while происходит обработка массива точек, сформированного алгоритмом обработки изображения. Условное выражение проверяет, является ли полученный массив – массивом точек от линии. Если условие удовлетворительное, то высчитывается погрешность для ПИД регулятора и задается новый угол движения, иначе робот продолжит двигаться в прежнем направлении.

```
def user_defined_followe_blue_line():
    global variable_lastMarker
    global variable_x
    global variable_cord
    global list_lastMarker
    global list_lineList
    global pid_Follow_line
    vision_ctrl.enable_detection(rm_define.vision_detection_line)
    vision_ctrl.line_follow_color_set(rm_define.line_follow_color_blue)
    vision_ctrl.enable_detection(rm_define.vision_detection_marker)
    vision_ctrl.set_marker_detection_distance(0.9)
    pid_Follow_line.set_ctrl_params(330,0,28)
    while True:
        led_ctrl.set_bottom_led(rm_define.armor_bottom_all, 36, 103, 255, rm_define.effect_always_on)
        list_lineList=RmList(vision_ctrl.get_line_detection_info())
        if len(list_lineList) == 42:
            if list_lineList[2] == 1:
                variable_x = list_lineList[19]
                pid_Follow_line.set_error(variable_x - 0.5)
                chassis_ctrl.move_with_speed(0.2,0,pid_Follow_line.get_output())
        else:
            chassis_ctrl.move_with_speed(0.1,0,pid_Follow_line.get_output())
```

Рис. 10. Движение по синей линии

Сканирование и анализ опознавательных меток происходит при помощи внутренних методов библиотеки robomaster SDK, т.к. разработчику предоставлен интерфейс взаимодействия. При распознавании опознавательного маркера с номером «3» (Рис. 11), тогда робот выполняет скрипт выгрузки объекта. При распознавании маркера с номером «1» робот следует по синей линии (Рис. 12), если маркер с номером «2», то по красной.

```
def vision_recognized_marker_number_three(msg):
    global variable_lastMarker
    global variable_x
    global variable_cord
    global list_lastMarker
    global list_lineList
    global pid_Follow_line
    led_ctrl.set_bottom_led(rm_define.armor_bottom_all, 100, 0, 100, rm_define.effect_always_on)
    chassis_ctrl.stop()
    time.sleep(2)
    chassis_ctrl.move_with_distance(0,0.4)
    chassis_ctrl.rotate_with_degree(rm_define.anticlockwise,90)
    user_defined_polozhit()

    chassis_ctrl.rotate_with_degree(rm_define.anticlockwise,90)
    chassis_ctrl.move_with_distance(0,0.1)
```

Рис. 11. Функция vision_recognized_marker_number_three();

```
def vision_recognized_marker_number_one(msg):
    global variable_lastMarker
    global variable_x
    global variable_cord
    global list_lastMarker
    global list_lineList
    global pid_Follow_line
    led_ctrl.set_bottom_led(rm_define.armor_bottom_all, 255, 193, 0, rm_define.effect_always_on)
    user_defined_in()

    user_defined_vzyt()

    user_defined_out()

    user_defined_follow_red_line()
```

Рис. 12. Функция vision_recognized_marker_number_one();

Взаимодействие с объектами происходит благодарят функциям «взять» (Рис. 13) и «положить» (Рис. 14) объект. Для этого следует управлять модулем захвата и положением плеч манипулятора. Следует учесть, что на эти действия требуется время, поэтому нужно ставить таймеры ожидания программы, чтобы механика успела завершить действие.

```
def user_defined_vzyt():
    global variable_lastMarker
    global variable_x
    global variable_cord
    global list_lastMarker
    global list_lineList
    global pid_Follow_line
    gripper_ctrl.open()
    time.sleep(2)
    robotic_arm_ctrl.move(100, 0, wait_for_complete=True)
    robotic_arm_ctrl.move(0, -80, wait_for_complete=True)
    chassis_ctrl.move_with_distance(0,0.1)
    gripper_ctrl.close()
    time.sleep(2)
    robotic_arm_ctrl.recenter(wait_for_complete=True)
    time.sleep(2)
    chassis_ctrl.move_with_distance(180,0.1)
```

Рис. 13. Функция user_defined_vzyt();

```
def user_defined_polozhit():
    global variable_lastMarker
    global variable_x
    global variable_cord
    global list_lastMarker
    global list_lineList
    global pid_Follow_line
    robotic_arm_ctrl.move(100, 0, wait_for_complete=True)
    robotic_arm_ctrl.move(0, -100, wait_for_complete=True)
    chassis_ctrl.move_with_distance(0,0.1)
    gripper_ctrl.open()
    time.sleep(2)
    chassis_ctrl.move_with_distance(180,0.1)
    robotic_arm_ctrl.recenter(wait_for_complete=True)
    gripper_ctrl.close()
    time.sleep(2)
```

Рис. 14. Функция user_defined_polozhit();

Данная программа работает только при выполнении требований, прописанных в Руководстве пользователя для настройки системы управления (см. приложение 1). С полным кодом программы можно ознакомиться в приложении (см. приложение 2).

2.3. Результат апробации, техническая документация

В ходе тестирования системы управления на тестовом поле была выявлена следующая проблема – нестабильность в поведении мобильного робота при движении по линии и активации режима поиска опознавательных меток. Данная проблема возникает из-за того, что разработчику в приложении

DJI RoboMaster не предоставляется возможность создавать асинхронные функции. Программа пишется в структурном стиле и функции работают последовательно. Это означает, что при движении по линии с последующим запуском скрипта, после сканирования метки, мобильный робот не сможет запустить следующий скрипт от сканирования другой метки. Подпрограмма функции отрабатывает в строгой последовательности и по завершении возвращается обратно к подпрограмме движения по линии. Данная особенность не позволяет создать сложный сценарий запуска подпрограмм на основе опознавательных меток и следует придерживаться правила: одна метка – одно подпрограмма первого уровня.

Для построения трассы под систему управления мобильным роботом следует обустраивать площадь от 5 $м^2$ с ситуативной постановкой перегородок. Широкий угол обзора камеры способствует ошибочным сканированиям соседних линий или меток, что приводит к нестабильному поведению мобильного робота.

По итогам тестирования системы управления мобильным роботом можно сделать вывод, что аппаратная часть мобильного робота DJI RoboMaser S1 требует большого пространства для корректной работы, а среда разработки в приложении DJI RoboMaster подходит, в большей степени, для обучения взаимодействию с роботом и базовым принципам работы технического зрения.

Для более сложных задач рекомендуется самостоятельно реализовывать программу и способы взаимодействия объектов друг с другом использую библиотеку robomaster SDK.

Заключение

Система управления мобильным роботом на основе технического зрения – актуальная и перспективная тема в робототехнике и программировании. Техническое зрение позволяет реализовать восприятие окружающей среды используя минимальное кол-во технического оснащении.

В ходе выполнения работы был проанализирован рынок существующих аппаратных решений мобильных роботов; изучены принципы ориентации роботов в пространстве и работы машинного зрения, изучены возможности мобильного робота RoboMaster S1, а также разработана и протестирована система управления с сопровождением технической документацией.

В ходе тестирования были выявленные проблемные места робота DJI RoboMaster S1, которые в будущем можно обходить при достаточном навыке разработки.

Дальнейшим направлением для данной дипломной работы является дополнение системы более сложными подпрограммами с помощью библиотеки robomaster SDK для разнообразия скриптов поведения робота.

Список информационных источников

1. Dajiang Innovation Technology Co RoboMaster S1 Руководство пользователя. - v 1.8 изд. - DJI, 2020. - 51 с.

2. Open Computer Vision Library // opencv.org URL: https://opencv.org/ (дата обращения: 05.03.2023).

3. Анализ поведения автоматических радиодальномеров при случайных возмущениях / В. В. Григорьев, Д. В. Козис, А. Н. Коровьяков, Ю. В. Литвинов // Изв.вузов. Приборостроение. — 2010. — № 7. — С. 26–32.

4. Власов С. М., Бойков В. И., Быстров С. В., Григорьев В. В. Бесконтактные средства локальной ориентации роботов. — СПб: Университет ИТМО, 2017. — 169с.

5. Давыденко Е.В. Разработка и анализ алгоритмов цифровой обработки сигналов в задаче оптической лазерной триангуляции: автореф. дис. канд. тех. наук: 05.12.04. - Владимир, 2009. - 24 с.

6. Искусство позиционирования // Журнал русского географического общества "Вокруг света" URL: https://www.vokrugsveta.ru/vs/article/2941/ (дата обращения: 04.03.2021).

7. Краткая история Python // skillbox.ru URL: https://skillbox.ru/media/code/kratkaya-istoriya-python/ (дата обращения: 29.04.2023).

8. НАУЧНО-ИССЛЕДОВАТЕЛЬСКАЯ РАБОТА «Исследование инерциальных систем навигации мобильных роботов» // sowa-ru.com URL: https://sowa-ru.com/wp-content/uploads/2020/09/lobova_ai_mht.b-61_2020_nir.pdf (дата обращения: 09.03.2023).

9. Порев В. Компьютерная графика. — СПб. : БХВ-Петербург, 2004.

10. Пример ультразвукового дальномера // Электронный ресурс. — Режим доступа: http://radio.delanet.ru.

11. Русинов М.М. Композиция оптических систем. - Ленинград: «Машиностроение»: Ленинградское отделение, 1989. - 379 с.

12. СИСТЕМА ТЕХНИЧЕСКОГО ЗРЕНИЯ ДЛЯ АВТОМАТИЧЕСКОГО ОРИЕНТИРОВАНИЯ И ПОЗИЦИОНИРОВАНИЯ МОБИЛЬНОГО РОБОТА // rusrobotics.ru URL: https://rusrobotics.ru/images/Archive_nomerov/1-2-2014/3-2-Yudin-Protsenko.pdf (дата обращения: 28.02.2023).

13. Цифровая обработка изображений в информационных системах: Учебное пособие. / И.С. Грузман, В.С. Киричук, В.П. Косых и др. — Новосибисрк : Изд-во НГТУ, 2002.

Приложения

Приложение 1.

Установка необходимого ПО, сборка, настройка, подключение и инструкция по эксплуатации DJI RoboMaster S1 описана в официальном руководстве пользования от DJI [1].

Для корректной работы системы управления мобильным роботом на основе технического зрения требуется строгая настройка опорных точек трассы:

- секция старта и финиша;
- секция поднятия объекта;
- секция выгрузки объекта;

Таблица 1.

Таблица размеров секций

Секция	**Размеры**
Старт и финиш (Start/Finish)	250x250 мм
Выгрузка объектов (P)	100x100 мм
Поднятие объектов («крест»)	80x80 мм

Размеры пути от стоп-линии до секций для текущей конфигурации системы управления изображены на схеме трассы (Рис. 1).

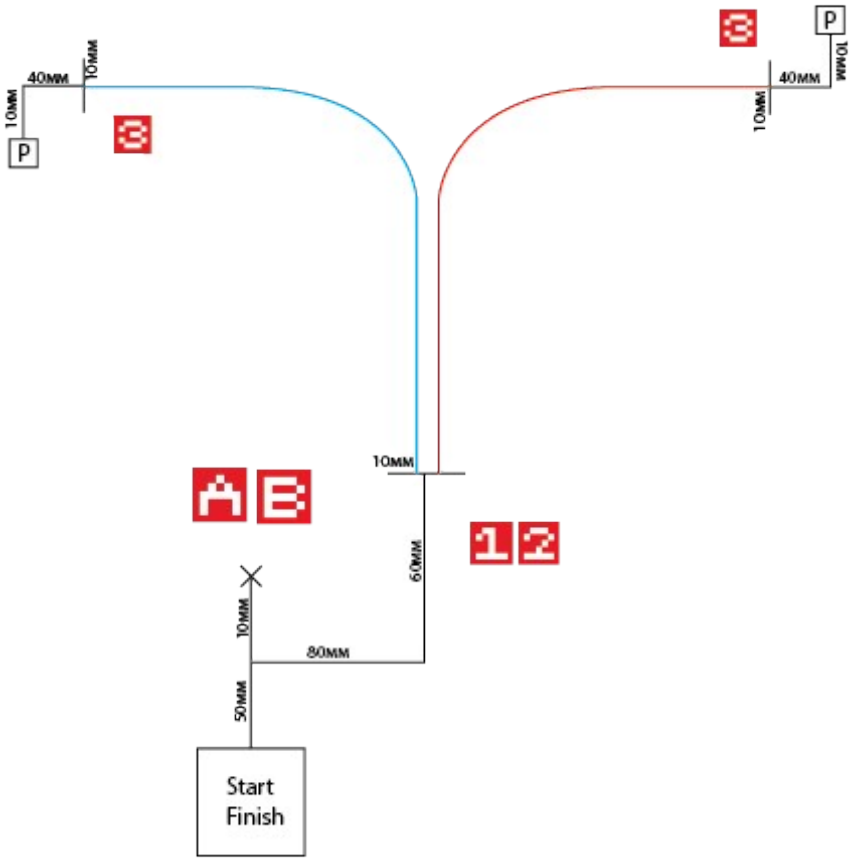

Рис. 1. Схема тестовой трассы

При редактировании линий скриптовых движений и секций следует менять параметры расстояния смещения и направления поворотов соответствующих методов: user_defined_in(), user_defined_out(), user_defined_start_work(), user_defined_finish_work(), vision_recognized_marker_number_one(), vision_recognized_marker_number_two(), vision_recognized_marker_number_three().

Расположение цветных линий не влияет на программу системы управления. При этом следует прокладывать маршрутные линии в соответствии с руководством пользователя DJI RoboMaster S1[1]

Приложение 2.

```
variable_lastMarker = 0
variable_x = 0
variable_cord = 0
list_lastMarker = RmList()
list_lineList = RmList()
pid_Follow_line = PIDCtrl()
def user_defined_out():
    global variable_lastMarker
    global variable_x
    global variable_cord
    global list_lastMarker
    global list_lineList
    global pid_Follow_line
    chassis_ctrl.rotate_with_degree(rm_define.clockwise,90)
    chassis_ctrl.move_with_distance(0,0.8)
    chassis_ctrl.rotate_with_degree(rm_define.anticlockwise,90)
    chassis_ctrl.move_with_distance(0,1.4)
def user_defined_in():
    global variable_lastMarker
    global variable_x
    global variable_cord
    global list_lastMarker
    global list_lineList
    global pid_Follow_line
    chassis_ctrl.stop()
    time.sleep(2)
    chassis_ctrl.set_trans_speed(0.2)
```

```
    chassis_ctrl.move_with_distance(0,0.6)
    chassis_ctrl.rotate_with_degree(rm_define.clockwise,90)
    chassis_ctrl.move_with_distance(0,0.8)
    chassis_ctrl.rotate_with_degree(rm_define.clockwise,90)
    chassis_ctrl.stop()
def user_defined_polozhit():
    global variable_lastMarker
    global variable_x
    global variable_cord
    global list_lastMarker
    global list_lineList
    global pid_Follow_line
    robotic_arm_ctrl.move(100, 0, wait_for_complete=True)
    robotic_arm_ctrl.move(0, -100, wait_for_complete=True)
    chassis_ctrl.move_with_distance(0,0.1)
    gripper_ctrl.open()
    time.sleep(2)
    chassis_ctrl.move_with_distance(180,0.1)
    robotic_arm_ctrl.recenter(wait_for_complete=True)
    gripper_ctrl.close()
    time.sleep(2)
def user_defined_vzyt():
    global variable_lastMarker
    global variable_x
    global variable_cord
    global list_lastMarker
    global list_lineList
    global pid_Follow_line
    gripper_ctrl.open()
```

```
    time.sleep(2)
    robotic_arm_ctrl.move(100, 0, wait_for_complete=True)
    robotic_arm_ctrl.move(0, -80, wait_for_complete=True)
    chassis_ctrl.move_with_distance(0,0.1)
    gripper_ctrl.close()
    time.sleep(2)
    robotic_arm_ctrl.recenter(wait_for_complete=True)
    time.sleep(2)
    chassis_ctrl.move_with_distance(180,0.1)
def user_defined_start_work():
    global variable_lastMarker
    global variable_x
    global variable_cord
    global list_lastMarker
    global list_lineList
    global pid_Follow_line
    chassis_ctrl.set_trans_speed(0.2)
    chassis_ctrl.move_with_distance(0,0.5)
    user_defined_out()

    chassis_ctrl.rotate_with_degree(rm_define.clockwise,180)
def user_defined_finish_work():
    global variable_lastMarker
    global variable_x
    global variable_cord
    global list_lastMarker
    global list_lineList
    global pid_Follow_line
    chassis_ctrl.stop()
```

```
        time.sleep(2)
        chassis_ctrl.set_trans_speed(0.2)
        chassis_ctrl.move_with_distance(0,0.6)
        chassis_ctrl.rotate_with_degree(rm_define.clockwise,90)
        chassis_ctrl.move_with_distance(0,0.8)
        chassis_ctrl.rotate_with_degree(rm_define.anticlockwise,90)
        chassis_ctrl.move_with_distance(0,0.5)
        chassis_ctrl.rotate_with_degree(rm_define.anticlockwise,180)
        chassis_ctrl.stop()
    def user_defined_followe_blue_line():
        global variable_lastMarker
        global variable_x
        global variable_cord
        global list_lastMarker
        global list_lineList
        global pid_Follow_line
        vision_ctrl.enable_detection(rm_define.vision_detection_line)
        vision_ctrl.line_follow_color_set(rm_define.line_follow_color_blue)
        vision_ctrl.enable_detection(rm_define.vision_detection_marker)
        vision_ctrl.set_marker_detection_distance(0.9)
        pid_Follow_line.set_ctrl_params(330,0,28)
        while True:
            led_ctrl.set_bottom_led(rm_define.armor_bottom_all, 36, 103, 255, rm_d
efine.effect_always_on)
            list_lineList=RmList(vision_ctrl.get_line_detection_info())
            if len(list_lineList) == 42:
                if list_lineList[2] == 1:
                    variable_x = list_lineList[19]
                    pid_Follow_line.set_error(variable_x - 0.5)
```

```
                chassis_ctrl.move_with_speed(0.2,0,pid_Follow_line.get_output())
            else:
                chassis_ctrl.move_with_speed(0.1,0,pid_Follow_line.get_output())
        def user_defined_follow_red_line():
            global variable_lastMarker
            global variable_x
            global variable_cord
            global list_lastMarker
            global list_lineList
            global pid_Follow_line
            vision_ctrl.enable_detection(rm_define.vision_detection_line)
            vision_ctrl.line_follow_color_set(rm_define.line_follow_color_red)
            vision_ctrl.enable_detection(rm_define.vision_detection_marker)
            vision_ctrl.set_marker_detection_distance(0.9)
            pid_Follow_line.set_ctrl_params(330,0,28)
            while True:
                led_ctrl.set_bottom_led(rm_define.armor_bottom_all, 255, 0, 0, rm_defin
e.effect_always_on)
                list_lineList=RmList(vision_ctrl.get_line_detection_info())
                if len(list_lineList) == 42:
                    if list_lineList[2] == 1:
                        variable_x = list_lineList[19]
                        pid_Follow_line.set_error(variable_x - 0.5)
                        chassis_ctrl.move_with_speed(0.2,0,pid_Follow_line.get_output())
                else:
                    chassis_ctrl.move_with_speed(0.1,0,pid_Follow_line.get_output())
        def start():
            global variable_lastMarker
            global variable_x
```

```
        global variable_cord
        global list_lastMarker
        global list_lineList
        global pid_Follow_line
        vision_ctrl.enable_detection(rm_define.vision_detection_marker)
        vision_ctrl.set_marker_detection_distance(1)
        media_ctrl.exposure_value_update(rm_define.exposure_value_medium)
        gripper_ctrl.close()
        robotic_arm_ctrl.recenter(wait_for_complete=True)
        user_defined_start_work()

        user_defined_follow_red_line()

        time.sleep(2)
        while True:
            pass
    def vision_recognized_marker_number_one(msg):
        global variable_lastMarker
        global variable_x
        global variable_cord
        global list_lastMarker
        global list_lineList
        global pid_Follow_line
        led_ctrl.set_bottom_led(rm_define.armor_bottom_all, 255, 193, 0, rm_defin
e.effect_always_on)
        user_defined_in()

        user_defined_vzyt()
```

```
            user_defined_out()

            user_defined_follow_red_line()

        def vision_recognized_marker_number_three(msg):
            global variable_lastMarker
            global variable_x
            global variable_cord
            global list_lastMarker
            global list_lineList
            global pid_Follow_line
            led_ctrl.set_bottom_led(rm_define.armor_bottom_all, 100, 0, 100, rm_defin
e.effect_always_on)
            chassis_ctrl.stop()
            time.sleep(2)
            chassis_ctrl.move_with_distance(0,0.4)
            chassis_ctrl.rotate_with_degree(rm_define.anticlockwise,90)
            user_defined_polozhit()

            chassis_ctrl.rotate_with_degree(rm_define.anticlockwise,90)
            chassis_ctrl.move_with_distance(0,0.1)
        def vision_recognized_marker_number_two(msg):
            global variable_lastMarker
            global variable_x
            global variable_cord
            global list_lastMarker
            global list_lineList
            global pid_Follow_line
```

led_ctrl.set_bottom_led(rm_define.armor_bottom_all, 255, 193, 0, rm_defin
e.effect_always_on)

user_defined_in()

user_defined_vzyt()

user_defined_out()

user_defined_followe_blue_line()