МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ РЕСПУБЛИКИ КАЗАХСТАН

Некоммерческое акционерное общество «Казахский национальный исследовательский технический университет имени К.И.Сатпаева»

SATBAYEV UNIVERSITY

Институт Автоматики и информационных технологий

Кафедра Робототехники и технических средств автоматики

Бейсултанов Адам Русланович

Тестирование алгоритмов управления в VR среде без необходимости физического прототипа

ПОЯСНИТЕЛЬНАЯ ЗАПИСКА
к дипломному проекту

6В07113 – Робототехника и мехатроника

Алматы 2025

МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ РЕСПУБЛИКИ КАЗАХСТАН

Некоммерческое акционерное общество «Казахский национальный исследовательский технический университет имени К.И.Сатпаева»

SATBAYEV UNIVERSITY

Институт Автоматики и информационных технологий

Кафедра Робототехники и технических средств автоматики

ДОПУЩЕН К ЗАЩИТЕ
Заведующий кафедрой РТиТСА
кандидат технических наук,
профессор
Ожикенов К. А.
«__» _____ 2025 г.

ПОЯСНИТЕЛЬНАЯ ЗАПИСКА
к дипломному проекту

На тему: «Тестирование алгоритмов управления в VR среде без необходимости физического прототипа »

6B07113 – Робототехника и мехатроника

Выполнил
Бейсултанов А.Р.

Научный руководитель
Магистр технических наук,
старший преподаватель
Байтурганова В.К.

«30» май 2025 г.

Рецензент
К. т. н., ассоциированный профессор
Сейдалиева А.К.

30 май 2025 г.

Алматы 2025

МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ РЕСПУБЛИКИ КАЗАХСТАН

Некоммерческое акционерное общество «Казахский национальный исследовательский технический университет имени К.И.Сатпаева»

SATBAYEV UNIVERSITY

Институт Автоматики и информационных технологий

Кафедра Робототехники и технических средств автоматики

6В07113 – Робототехника и мехатроника

УТВЕРЖДАЮ
Заведующий кафедрой РТиТСА
кандидат технических наук,
профессор
_____ Ожикенов К. А.
«__» _____ 2025 г.

ЗАДАНИЕ
на выполнение дипломного проекта

Студенту:Бейсултанов Адам Русланович
Тема: Тестирование алгоритмов управления в VR среде без необходимости физического прототипа
Утверждена приказом ректора_____ №521 от «13» 11 2024 г.
Срок сдачи законченной работы «30» 05 2025 г.
Исходные данные к дипломному проекту: (законы, литературные источники, лабораторно-производственные данные)
Теоретические материалы по использованию VR
Практические данные для тестирования алгоритмов
Теоретические данные по использованию CompeliaSim
Перечень подлежащих разработке в дипломном проекте вопросов:
1.Анализ применения VR-технологий для тестирования алгоритмов управления.
2.Разработка виртуальной среды для имитации процесса управления.
3.Реализация и тестирование алгоритмов управления в VR.

Перечень графического материала (с точным указанием обязательных чертежей): представлены слайдов презентации работы)
Рисунки:35
Таблицы:5

ГРАФИК
подготовки дипломной работы (проекта)

Наименование разделов, перечень разрабатываемых вопросов	Сроки представления научному руководителю	Примечания
Теоретическая часть	15.01-21.02.2025 г.	Выполнено
Практическая часть	23.02-08.04.2025 г.	Выполнено
Подготовка документаций	08.04-25.05.2025 г	Выполнено

Подписи
консультантов и норм контролера на законченную дипломную работу (проект)с указанием относящихся к ним разделов работы (проекта)

Наименование разделов	Консультанты, И.О.Ф. (уч. степень, звание)	Дата подписания	Подпись
Нормоконтролер	Кальменов Е.Т. магистр технических наук, старший преподаватель	05.06.2025	

Научный руководитель: Байтурганова В.К.

Задание принял к исполнению обучающийся: Бейсултанов А.Р.

Дата «5» 06 2025

АННОТАЦИЯ

В этой дипломной работе рассматривается, как технологии виртуальной реальности (VR) могут использоваться для тестирования и улучшения алгоритмов управления роботами. Современная робототехника требует высокой гибкости и адаптивности, поэтому виртуальная среда становится отличным инструментом: она позволяет безопасно и эффективно моделировать сложные ситуации, в которых может оказаться робот, не тратя ресурсы на создание физических прототипов.Работа состоит из двух частей — теоретической и практической. В теоретической части проанализированы существующие подходы и VR-технологии, которые уже применяются в робототехнике. Также рассматриваются их преимущества по сравнению с традиционными методами — например, возможность быстро создавать и изменять сценарии, использовать обучение с подкреплением и оптимизировать работу управляющих алгоритмов.Практическая часть включает в себя серию экспериментов в виртуальной среде CoppeliaSim. В ходе этих экспериментов моделировались разные ситуации, в которых может работать робот. Это помогло на практике оценить, как различные параметры и условия влияют на эффективность алгоритмов управления.В результате работы стало очевидно, что использование VR значительно ускоряет процесс разработки и позволяет добиться более высокого качества решений. Основной вывод: интеграция виртуальной реальности в процессы создания и тестирования роботов не только делает их разработку удобнее, но и открывает новые возможности для инноваций в этой области.

ABSTRACT

This thesis examines how virtual reality (VR) technologies can be used for testing and improving robot control algorithms. Modern robotics requires high flexibility and adaptability, so the virtual environment becomes an excellent tool: it allows safe and efficient modeling of complex situations in which a robot may find itself, without spending resources on creating physical prototypes. The work consists of two parts — theoretical and practical. The theoretical part analyzes existing approaches and VR technologies that are already applied in robotics. Their advantages compared to traditional methods are also considered — for example, the ability to quickly create and modify scenarios, use reinforcement learning, and optimize the operation of control algorithms. The practical part includes a series of experiments in the virtual environment CoppeliaSim. During these experiments, different situations in which a robot can operate were simulated. This helped practically assess how various parameters and conditions affect the efficiency of control algorithms. As a result of the work, it became clear that using VR significantly speeds up the development process and allows achieving higher quality solutions. The main conclusion: integrating virtual reality into the processes of creating and testing robots not only makes their development more convenient but also opens new opportunities for innovation in this field.

АҢДАТПА

Бұл дипломдық жұмыста виртуалды шындық (VR) технологияларының роботтарды басқару алгоритмдерін тестілеу және жақсарту үшін қалай қолданылатыны қарастырылады. Қазіргі робототехника жоғары икемділік пен бейімделушілікті талап етеді, сондықтан виртуалды орта күрделі жағдайларды қауіпсіз әрі тиімді модельдеуге мүмкіндік беретін керемет құралға айналады, ол үшін физикалық прототиптер жасауға көп ресурс қажет емес.Жұмыс екі бөлімнен тұрады — теориялық және практикалық. Теориялық бөлімде робототехникада қолданылып жүрген VR технологиялары мен қазіргі тәсілдер талданады. Сонымен қатар, олар дәстүрлі әдістерге қарағанда артықшылықтары қарастырылады — мысалы, сценарийлерді жылдам жасау және өзгерту, күшейтумен оқыту әдістерін қолдану және басқару алгоритмдерін оңтайландыру мүмкіндіктері.Практикалық бөлімде CoppeliaSim виртуалды ортасында бірнеше тәжірибе жүргізілді. Бұл тәжірибелерде робот жұмыс істеуі мүмкін әртүрлі жағдайлар модельденді. Нәтижесінде әртүрлі параметрлер мен жағдайлардың басқару алгоритмдерінің тиімділігіне қалай әсер ететіні тәжірибеде бағаланды.Жұмыстың қорытындысы бойынша VR технологияларын қолдану әзірлеу үдерісін едәуір жылдамдатып, шешімдердің сапасын жоғарылатуға мүмкіндік береді деп анықталды. Негізгі тұжырым — виртуалды шындықты роботтарды жасау және тестілеу үдерісіне енгізу олардың дамуын жеңілдетіп қана қоймай, бұл салада инновацияларға жаңа жол ашады.

СОДЕРЖАНИЕ

ВВЕДЕНИЕ

Современные технологии развиваются с невероятной скоростью, и виртуальная реальность (VR) уверенно занимает важное место среди передовых направлений. Особенно ярко её влияние проявляется в таких сферах, как робототехника. За последние несколько лет мы видим, как VR всё активнее внедряется в процессы проектирования, тестирования и совершенствования роботизированных систем. Почему так происходит? Всё просто: традиционные методы разработки требуют много времени, сил и ресурсов. Создание физических прототипов — дело не только затратное, но и не всегда оправданное, особенно на начальных этапах. А вот виртуальная реальность позволяет моделировать работу робота в безопасной, полностью управляемой среде.Разработчики могут проверять и оттачивать алгоритмы управления прямо на виртуальной модели — без риска и без необходимости собирать дорогие прототипы. Более того, VR даёт возможность проводить испытания в условиях, которые в реальности воспроизвести либо крайне сложно, либо вообще невозможно — например, в экстремальной жаре или холоде, в космосе, под водой и так далее.Таким образом, виртуальная реальность открывает перед робототехникой совершенно новые горизонты, делая процесс разработки быстрее, безопаснее и эффективнее.

Цель этой работы — разобраться, как технологии виртуальной реальности могут использоваться для тестирования и улучшения алгоритмов управления роботами в разных условиях. Чтобы этого добиться, нужно решить несколько важных задач: изучить уже существующие VR-системы и примеры их использования в робототехнике, разработать понятную методику создания виртуальных сред для симуляции, а также создать прототип VR-приложения, в котором можно будет тестировать алгоритмы управления.В процессе исследования будет проанализировано, каким образом виртуальная реальность способна упростить и ускорить разработку роботизированных систем, сделать их более эффективными и в то же время сократить расходы — в частности, за счёт уменьшения потребности в создании дорогостоящих физических моделей на ранних этапах. Ожидается, что результаты этой работы будут полезны не только учёным, но и практикам — тем, кто занимается робототехникой на прикладном уровне и стремится внедрять современные, продвинутые технологии в свою работу.

Актуальность использования технологий виртуальной реальности для тестирования и оптимизации алгоритмов управления роботами в последние годы значительно возросла. Это связано с рядом важных факторов, которые ясно

показывают, почему VR становится всё более необходимой в процессе разработки и тестирования.Во-первых, стоит обратить внимание на стремительный рост рынка VR/AR-технологий. По прогнозам, к 2025 году его объём может достичь 65,9 миллиардов долларов (Рис. 1). Это говорит о том, что подобные технологии находят всё более широкое применение в самых разных сферах, включая промышленность и робототехнику. Такой рост создаёт сильный стимул для компаний инвестировать в VR-решения, поскольку они открывают новые пути для повышения эффективности работы и ускорения разработки.

Рисунок 1 - Динамика мирового рынка AR/VR с 2020 по 2025 источник [12]

Во-вторых, виртуальное моделирование даёт возможность создавать очень реалистичные симуляции, в которых можно заранее просматривать, как робот будет взаимодействовать с окружающей средой. Это особенно полезно в тех случаях, когда реальные условия либо опасны, либо просто недоступны. Благодаря VR можно на этапе проектирования заметить и устранить потенциальные проблемы, а все элементы системы — протестировать до начала сборки настоящего устройства. Такой подход действительно помогает сэкономить и время, и ресурсы.В-третьих — и это, пожалуй, одно из главных преимуществ — VR можно использовать там, где человеку находиться небезопасно. С помощью виртуальной среды инженеры могут управлять роботами дистанционно, например, на объектах с высокими температурами, в зонах с химической или радиационной угрозой. Это не только снижает риски для человека, но и даёт возможность протестировать, как робот будет вести себя в самых сложных условиях. Такие технологии уже находят применение на опасных производствах, вроде металлургии, или при работе в заражённых зонах. Тут

виртуальная реальность становится по-настоящему важным инструментом, который помогает обеспечивать безопасность.

Всё это говорит о том, что тема этой работы действительно актуальна — сегодня как никогда важно исследовать, как VR может помочь сделать разработку и тестирование роботов быстрее, безопаснее и дешевле. В рамках проекта планируется использовать CoppeliaSim — это удобная программа, в которой можно создавать виртуальные модели роботов и «прокручивать» разные сценарии их поведения. Такой подход не только ускоряет работу над проектом, но и позволяет сделать алгоритмы управления точнее и надёжнее, ведь всё тестируется в чётко контролируемых условиях.

2. История развития отрасли

2.1 Виртуальная реальность основные понятия

Виртуальная реальность (VR) — это технология, которая позволяет создавать искусственную трёхмерную среду, в которую пользователь может буквально «погрузиться» и взаимодействовать с её элементами так, как если бы всё происходило в реальной жизни. С помощью VR человек может перемещаться по вымышленным пространствам, исследовать объекты и выполнять разные действия — от простых манипуляций до сложных сценариев взаимодействия [1]. Чтобы это погружение в виртуальный мир было максимально полным, нужны определённые устройства. В первую очередь — VR-гарнитура. Это основное средство визуализации, которое надевается на голову и позволяет «увидеть» виртуальную среду в объёме. Гарнитуры бывают разные: одни работают автономно (например, Oculus Quest), другие подключаются к компьютеру (такие как HTC Vive или Oculus Rift). Они оснащены экранами с высоким разрешением и датчиками, которые отслеживают движения головы, обеспечивая эффект присутствия. Контроллеры — это то, что позволяет управлять виртуальной средой: брать и перемещать объекты, нажимать кнопки, взаимодействовать с элементами пространства [1]. Чтобы пользователь чувствовал себя в виртуальном мире естественно, также применяются сенсоры движения — они отслеживают перемещения тела и рук в пространстве, создавая более реалистичную картину происходящего (Рис. 2.1).

За визуализацией и логикой взаимодействий стоит программное обеспечение — такие среды, как CoppeliaSim, Unity или Unreal Engine. Именно с их помощью создаются виртуальные миры, задаются сценарии и реализуется взаимодействие между пользователем и объектами.Основой работы VR являются несколько ключевых принципов. Прежде всего — **иммерсивность**, то есть эффект полного погружения. Он достигается за счёт объёмного изображения и пространственного звука, которые заставляют пользователя чувствовать себя частью другого мира. Важна также **интерактивность** — возможность напрямую влиять на происходящее с помощью контроллеров, движений или других форм ввода.Отдельно стоит отметить **трекинг** — это система, которая в реальном времени отслеживает движения человека и адаптирует виртуальную картинку под его действия. Благодаря этому взаимодействие с VR становится максимально естественным. Сфера применения VR невероятно широка. В игровой индустрии создаются уникальные игры с полным погружением. В образовании — интерактивные классы и симуляции, которые помогают ученикам осваивать

сложные темы «вживую». В медицине VR используется для тренировки врачей, проведения терапий (например, при фобиях) и даже в облегчении боли. Архитекторы применяют VR для просмотра своих проектов в 3D ещё до начала строительства. А в робототехнике виртуальная реальность — настоящий прорыв. Она даёт возможность тестировать и оттачивать алгоритмы управления без необходимости собирать физические прототипы. Это не только ускоряет процесс, но и делает его значительно дешевле и безопаснее [1].

Рисунок 2-Компоненты VR источник [13]

2.2 Технологии виртуальной реальности в робототехнике

Виртуальная реальность (VR) — это технология, которая создаёт искусственную среду, с которой пользователь может взаимодействовать с помощью специальных устройств: гарнитур, контроллеров, сенсоров и другого оборудования. За последние годы VR всё активнее находит применение в робототехнике, открывая разработчикам новые возможности для моделирования, тестирования и улучшения роботизированных систем. Одним из основных инструментов в этой области стала программа CoppeliaSim (ранее известная как V-REP). Это мощная платформа, позволяющая создавать виртуальные модели роботов и проводить их симуляцию в реалистичной среде. Используя её, можно не только визуализировать работу системы, но и заранее протестировать её поведение в разных условиях, не прибегая к созданию физического прототипа. Важно понимать базовые принципы работы VR [2]. Сама по себе технология строится на создании трёхмерного пространства, в котором можно не просто «присутствовать», а полноценно взаимодействовать с объектами. В сочетании с искусственным интеллектом VR становится ещё более мощным инструментом: роботы могут анализировать огромное количество данных, учиться, принимать

решения и выполнять задачи, которые для человека были бы слишком сложными или рискованными.Такой подход позволяет не только ускорить обработку данных, но и сократить количество операторов, нужных для управления системой. Робот, работающий в VR-среде, способен максимально эффективно выполнять свою задачу — при этом без вреда для окружающей среды и без рисков для человека.Сегодня VR уже используется для решения множества задач: от мониторинга и управления безопасностью до настройки навигационных систем, машинного обучения и анализа информации. Это помогает оптимизировать работу роботов и добиться более высокой производительности. За счёт своей гибкости VR становится незаменимым инструментом при разработке, тестировании и обучении роботизированных систем [1].Использование VR даёт разработчикам возможность в буквальном смысле «поиграться» с моделью робота, не тратя ресурсы на его физическую сборку. Это особенно ценно на этапе проектирования, когда каждая ошибка может обойтись дорого и в прямом, и в переносном смысле [2] [3].

Примеры приминения: Telexistence TX SCARA (США, Япония):Эта система сочетает в себе VR, ИИ и робототехнику для автоматического пополнения полок в супермаркетах. Управление осуществляется через VR-интерфейс, и оператор может вмешаться дистанционно, если что-то пошло не так.

VRoxy для удалённого управления (США): Система VRoxy даёт возможность оператору управлять удалённым роботом с помощью VR-гарнитуры. Это особенно полезно в медицине, промышленности и научных исследованиях, где важна высокая точность движений. Даже мелкая моторика может быть передана роботу через интерфейс с минимальной задержкой.

Проблема	Описание	
Ограниченность применения		Трудности в применении VR-технологий в промышленности заключаются в их ограниченности применения.
Дорогостоящая инфраструктура		Для эффективного использования VR технологий необходима дорогостоящая инфраструктура.
Затраты на разработку		Разработка программного обеспечения и инструментов для VR-технологий очень дорогостоящая.
Недостаточный уровень развития		VR еще недостаточно развита для промышленной робототехники и дает недостаточный потенциал для решения различных проблем.

Таблица 1 - Проблемы использования VR технологий

Конечно, нельзя говорить только о плюсах. У VR-технологий есть и свои минусы, которые могут серьёзно повлиять на результаты работы. Высокая стоимость оборудования, требовательность к ресурсам компьютера, возможность ошибок в виртуальной симуляции — всё это важно учитывать. Именно поэтому в своём исследовании я хочу сосредоточиться не только на общих возможностях VR, но в первую очередь — на проектировании виртуальных моделей роботов, их поведении в разных условиях, а также на тестировании и отладке алгоритмов управления. Особенно важно понять, как робот будет действовать в сложных сценариях — от преодоления препятствий до работы в условиях высокой или низкой температуры. Такой подход позволяет

глубже понять, где VR действительно помогает, а где его возможности пока ограничены.

2.3 Симбиоз VR и исскуственного интелекта

Искусственный интеллект — это то, что позволяет роботам думать, учиться на собственном опыте и принимать осознанные решения. Благодаря методам глубокого обучения роботы умеют распознавать объекты, строить планы действий и даже предсказывать возможные сценарии развития событий. Виртуальная реальность активно используется для создания симуляций, в которых роботы могут «обучаться» в безопасной и контролируемой среде. Здесь моделируются самые разные ситуации — от заводских процессов до чрезвычайных спасательных операций. Робот отрабатывает действия, учится взаимодействовать с предметами и людьми, и затем переносит эти навыки в реальный мир [5]. Симбиоз VR (виртуальной реальности) и ИИ (искусственного интеллекта) представляет собой мощное сочетание технологий, которое открывает новые возможности в различных областях. VR обеспечивает захватывающую и интерактивную среду, а ИИ добавляет интеллект, адаптивность и персонализацию [2]. Вместе они создают возможности, которые ранее были невозможны.

Области применения и примеры: Игры и развлечения: ИИ способен создавать реалистичные и постоянно меняющиеся игровые миры. NPC (неигровые персонажи) могут реагировать на действия игрока, запоминать их и подстраивать сюжет под конкретный стиль игры. Представьте VR-игру, где сюжет формируется индивидуально под каждого пользователя. Обучение и симуляция: VR и ИИ позволяют создавать продвинутые симуляторы для тренировки в сложных и опасных условиях. Врачи, например, могут отрабатывать хирургические операции в VR, а ИИ будет анализировать их действия и давать обратную связь. Пилоты, солдаты, пожарные — все могут проходить реалистичное обучение без риска для жизни.Образование: VR делает обучение более захватывающим, а ИИ помогает адаптировать учебный процесс под каждого ученика [4]. Например, ученик может участвовать в VR-уроке истории, где он перемещается во времени и взаимодействует с историческими событиями и персонажами.

Здравоохранение: VR активно используется для лечения тревожных расстройств, фобий и ПТСР. ИИ при этом отслеживает поведение пациента в виртуальной среде и подбирает подходящие терапевтические упражнения. Также такие технологии применяются в реабилитации после инсультов и травм — ИИ

анализирует прогресс и корректирует курс восстановления. Проектирование и инженерия: VR позволяет конструкторам и инженерам работать с 3D-моделями в масштабе один к одному. ИИ при этом может анализировать дизайн, выявлять слабые места и предлагать улучшения. Архитекторы, например, могут использовать VR для визуализации зданий, а ИИ — для расчёта энергоэффективности и оптимальной планировки.Искусство и творчество: VR открывает новые горизонты для художников, позволяя создавать иммерсивные проекты [6]. ИИ может предлагать идеи, генерировать элементы контента или адаптировать произведения под вкус конкретного зрителя. Представьте VR-галерею, где картины формируются в реальном времени в зависимости от настроения зрителя.Доступность: VR и ИИ делают цифровую среду более удобной для людей с ограниченными возможностями. VR создаёт адаптированные пространства, а ИИ может преобразовывать текст в речь и наоборот — в реальном времени.

Будущее симбиоза VR и ИИ:С каждым годом технологии становятся всё теснее взаимосвязаны. В будущем нас ждут ещё более реалистичные VR-среды, более интеллектуальные NPC и максимально персонализированные интерфейсы. Такой симбиоз способен радикально изменить то, как мы работаем, учимся, играем и взаимодействуем с окружающим миром.Симбиоз виртуальной реальности (VR) и искусственного интеллекта (ИИ) в робототехнике представляет собой мощное сочетание технологий, которое позволяет создавать более умные, адаптивные и эффективные системы [6]. Эти две области открывают новые возможности для того, чтобы роботы могли лучше понимать окружающий мир, взаимодействовать с ним и выполнять сложные задачи.

Преимущества	Применение
Адаптивность: Роботы с ИИ способны подстраиваться под изменения в окружающей среде. Если что-то идет не так на производственной линии, робот может самостоятельно найти решение. Эффективность: Использование VR ускоряет процесс обучения роботов. Вместо того чтобы тратить месяцы на тесты в реальной жизни, можно провести тысячи симуляций за считанные дни.	Медицина: Хирургические роботы обучаются с помощью VR-симуляций, что позволяет им выполнять сложные операции с высокой точностью. Промышленность: Роботы на заводах используют ИИ для оптимизации работы конвейеров, а VR помогает им учиться работать с новыми материалами или инструментами.
Безопасность: VR позволяет тестировать роботов в сложных или опасных условиях без риска для людей.	Спасательные операции: Роботы с ИИ и VR-тренировкой могут исследовать завалы после землетрясений или работать в условиях радиации.

Таблица 2 - Приемущества использования VR и ИИ

В итоге симбиоз VR и ИИ сделает роботов ещё более автономными и «умными». Они смогут не только выполнять поставленные задачи, но и самостоятельно предлагать решения для сложных проблем. Виртуальная реальность станет ещё более реалистичной, что даст роботам возможность тренироваться в условиях, максимально приближённых к реальным. В будущем возможно появление полностью автономных систем, которые смогут учиться и развиваться без постоянного участия человека [2].

2.4 Применение VR для тестирования алгоритмов управления

Алгоритмы — это чётко прописанные последовательности действий, которые нужны для решения конкретных задач. Их можно представить в виде текста, схемы или программы, и они широко применяются в разных сферах — от математики и информатики до робототехники. В робототехнике алгоритмы играют особенно важную роль, так как именно они управляют движениями и поведением роботов.Существует несколько видов алгоритмов, каждый из

которых имеет свои особенности и сферы применения [7]:Линейные алгоритмы: эти алгоритмы выполняют действия последовательно, шаг за шагом. Например, если роботу нужно пройти по определённому маршруту, он просто последовательно выполнит все инструкции от начала до конца, без каких-либо вариантов и отклонений (рис. 2.4.2) [7]. Разветвляющиеся алгоритмы: в таких алгоритмах есть выбор в зависимости от условий. Например, если робот встречает препятствие, он может принять решение пойти тем или иным путём, опираясь на данные, полученные с сенсоров.

Циклические алгоритмы: эти алгоритмы повторяют определённые действия, пока не выполнится нужное условие. К примеру, робот может ехать вперёд, пока не столкнётся с преградой [7]. Рекурсивные алгоритмы: здесь функция вызывает сама себя для решения частей задачи. Это удобно, когда большую задачу можно разбить на похожие, но более простые задачи — например, при прохождении лабиринта [7]. Генетические алгоритмы: эти алгоритмы вдохновлены природным отбором и применяются для поиска оптимальных решений в сложных задачах. Например, с их помощью можно найти лучший маршрут или оптимальную конфигурацию робота [8]. Алгоритмы машинного обучения: они дают роботам возможность учиться на основе данных и опыта

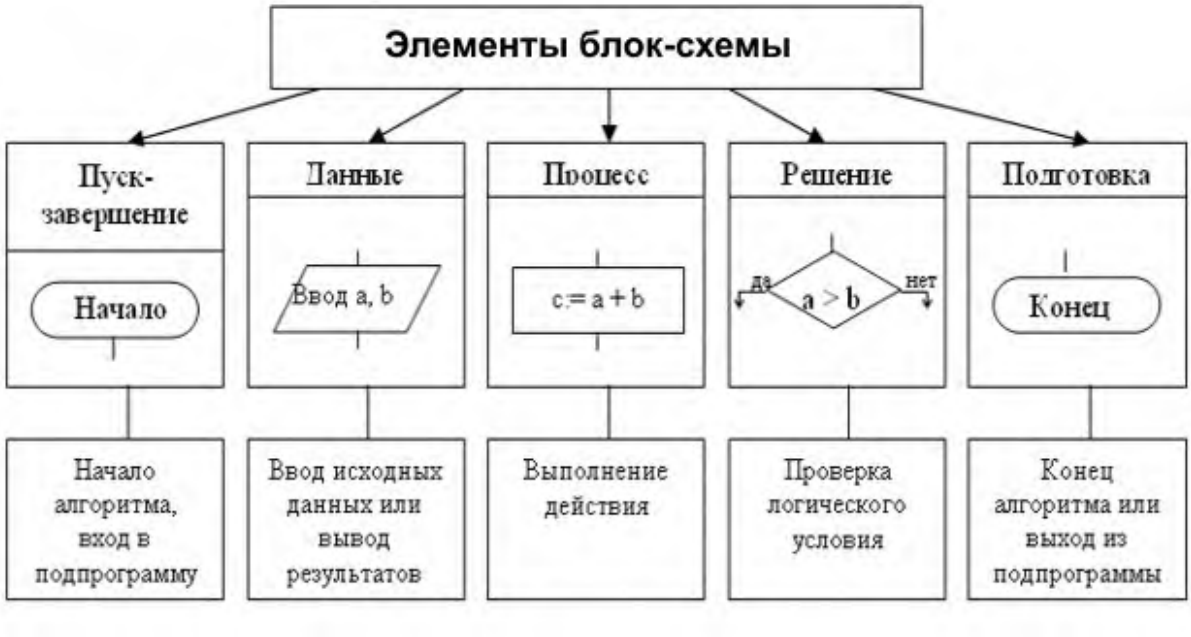


Рисунок 3-Упрощенный вид функциональных элементов блок схемы алгоритма источник [7]

Рисунок 4 -Простейшие примеры линейных алгоритмов источник [7]

Алгоритмы управления в робототехнике — это набор инструкций или последовательность действий, которые позволяют роботу выполнять определённые задачи и взаимодействовать с окружающей средой. Эти алгоритмы обеспечивают возможность достижения поставленных целей и могут варьироваться от простых до сложных [7]. Основные задачи алгоритмов управления включают планирование движений, которое определяет траекторию движения робота с учетом препятствий, а также планирование положений, сил и моментов, необходимых для выполнения задач [3]. Применение виртуальной реальности виртуальной реальности при тестировании алгоритмов управления в робототехнике открывает новые возможности для разработки и оптимизации роботизированных систем. VR делает возможными более реалистичные симуляции, которые могут использоваться для тестирования и отладки алгоритмов [2]. Это позволяет сократить затраченное время и обработать большее количество партий алгоритмов перед их физическим тестированием, что может подвергнуть системе опасность и неудачи. Можно выявить основные направления алгоритмов управления: эмуляция окружающей среды. Одно из преимуществ тестирования алгоритмов в VR состоит в возможности создания трехмерных моделей самой рабочей среды с роботами, допустим. Здесь можно изучать, как поступала бы машина в том же плане, но уже на физических объектах. Алгоритмы управления манипуляторами могут быть протестированы в VR-среде, где робот должен захватывать и перемещать объекты [8]. Например, разработчики могут использовать VR для моделирования различных сценариев

захвата, таких как перемещение предметов различной формы и веса. Это помогает оптимизировать алгоритмы, чтобы они могли точно выполнять задачи в реальном мире. Виртуальная реальность предоставляет возможность тестировать алгоритмы следования за движущимися объектами. Например, робот может быть запрограммирован следовать за мячом или другим объектом в виртуальной среде. Разработчики могут настраивать параметры алгоритма, чтобы улучшить его реакцию на изменения скорости и направления объекта [6]. Примеры алгоритмов управления. Алгоритмы стайного децентрализованного управления используются для управления группой роботов (например, дронов или колесных роботов), которые взаимодействуют друг с другом, обмениваясь данными. Такие алгоритмы основаны на методе искусственных сил, где каждый робот реагирует на виртуальные силы от соседей или окружающих объектов. Это позволяет группе двигаться как единое целое, избегая столкновений и выполняя задачи, такие как сбор ресурсов или достижение целевой точки [9]. Алгоритмы управления манипулятором через интернет включают управление роботами-манипуляторами через сеть с использованием VR. Виртуальная среда позволяет оператору видеть трехмерную модель рабочего пространства робота в реальном времени. Это особенно полезно для захвата подвижных или сложных объектов, где требуется высокая точность. Алгоритмы учитывают задержки связи и оптимизируют управление для работы даже при низкой скорости соединения [9]. Алгоритмы автономного движения применяются для роботов, которые должны перемещаться по заданным маршрутам или выполнять задачи в замкнутых пространствах. Например: следование по линии с использованием датчиков, выход из лабиринта с применением правил "правой" или "левой руки", обход препятствий с помощью сенсоров [9]. Алгоритмы последовательного упрощения моделей используются для решения задач кинематики роботов (прямой и обратной). Эти алгоритмы позволяют рассчитывать траектории движения звеньев манипуляторов или колесных роботов с минимальными вычислительными затратами, что важно для реального времени [5].

Алгоритмы управления промышленными роботами включают онлайн- и офлайн-программирование для выполнения задач на производстве. Онлайн-программирование подразумевает непосредственное управление роботом в реальном времени, а офлайн-программирование — создание траекторий и их тестирование в симуляциях до запуска на реальном оборудовании. Виртуальная реальность (VR) активно используется для тестирования алгоритмов управления в робототехнике, предоставляя возможность создавать безопасные симуляции для проверки и оптимизации поведения роботов [2]. Например, в VR можно

смоделировать лабиринт, где робот, используя алгоритмы обхода препятствий, находит выход. Это позволяет разработчикам тестировать навигационные стратегии без риска повреждения оборудования. Также VR применяется для отработки манипуляций: робот-манипулятор может захватывать и перемещать виртуальные объекты, что помогает улучшить точность и надежность алгоритмов захвата. Другой пример — тестирование алгоритмов следования за объектами, где робот в виртуальной среде учится реагировать на изменения скорости и направления движущихся объектов [10]. Кроме того, VR используется для телеуправления: операторы управляют роботами через виртуальные интерфейсы, наблюдая за их действиями в режиме реального времени, что особенно полезно в сложных или опасных условиях. VR также применяется для обучения алгоритмов машинного обучения, предоставляя роботу возможность адаптироваться к различным условиям окружающей среды на основе данных, собранных в симуляциях [10]. Еще одним примером является моделирование взаимодействия робота с пользователем, что помогает оптимизировать алгоритмы управления для более естественного взаимодействия. Таким образом, VR позволяет тестировать и совершенствовать алгоритмы управления роботов в безопасной и контролируемой среде.

Вывод: стоит подчеркнуть, что алгоритмы — это своего рода «мозг» любой робототехнической системы. Именно они задают логику поведения, позволяют роботам учиться на опыте, быстро реагировать на изменения окружающей среды и принимать решения, которые раньше были доступны только человеку. Благодаря развитию алгоритмов современные роботы способны не просто выполнять заранее заданные инструкции, а самостоятельно анализировать сложные ситуации, взаимодействовать с другими устройствами и даже предугадывать последствия своих действий. Всё это делает роботов неотъемлемой частью нашей жизни и открывает новые горизонты для их применения в самых разных сферах — от промышленности и медицины до образования и быта.

2.5 Основные методы оптимизации алгоритмов управления

Оптимизация алгоритмов управления — ключевой момент для создания эффективных систем в робототехнике, автоматике и информатике. Цель — повысить производительность, снизить затраты и повысить надёжность [9]. Современные системы сложны, поэтому важны разные методы оптимизации: параметрическая, линейное и нелинейное программирование, эвристики и аналитика, чтобы адаптировать алгоритмы под задачи. Параметрическая оптимизация выбирает лучшие параметры алгоритма, часто с помощью метода Монте-Карло, который хорош для линейных систем. Линейное программирование решает задачи с линейными функциями и ограничениями, используя, например, симплекс-метод [10]. Нелинейное программирование сложнее из-за локальных минимумов. Оптимизация бывает безусловной — без ограничений, и условной — с ограничениями, которые учитываются с помощью множителей Лагранжа или штрафных функций. Эвристические методы — генетические алгоритмы, рой частиц — помогают решать сложные задачи с множеством локальных минимумов. Численные методы — приближённые алгоритмы типа координатного спуска и случайного поиска. Аналитические — на основе производных, например, метод наискорейшего спуска и Ньютона. В робототехнике применяются все эти методы, а также специализированные.

Модельно-предиктивное управление (MPC) оптимизирует траектории робота в реальном времени с учётом ограничений и препятствий. Коллективное управление оптимизирует действия группы роботов, снижая нагрузку на центральный процессор. Двухуровневые интеллектуальные системы планирования работают на грубом и точном уровнях, иногда с генетическими алгоритмами или нечеткой логикой. Алгоритмы роя частиц (PSO) помогают координировать движения группы роботов, оптимизируя путь[10]. Численные методы используются для робастного управления, а выпуклое программирование — для эффективного управления шагающими роботами. Для автономного движения робота обычно делают двухэтапное планирование: глобальное — прокладка пути с обходом препятствий, и локальное — сглаживание с учётом кинодинамики[10]. MPC позволяет адаптироваться к изменениям в реальном времени, добавляя штрафы за столкновения через искусственные потенциальные поля (APF). Интересный подход — нейронное потенциальное поле (NPField), вдохновлённое моделями машинного зрения NeRF. NPField использует нейросеть для оценки отталкивающего потенциала и

его градиентов, чтобы оптимизировать траекторию [11]. Контроллер включает блоки для MPC и численного решения, обучается на данных о позе робота и картах препятствий. Такой подход помогает эффективно избегать столкновений и корректировать путь в реальном времени [11]. Схема работы показана на рисунке ниже:

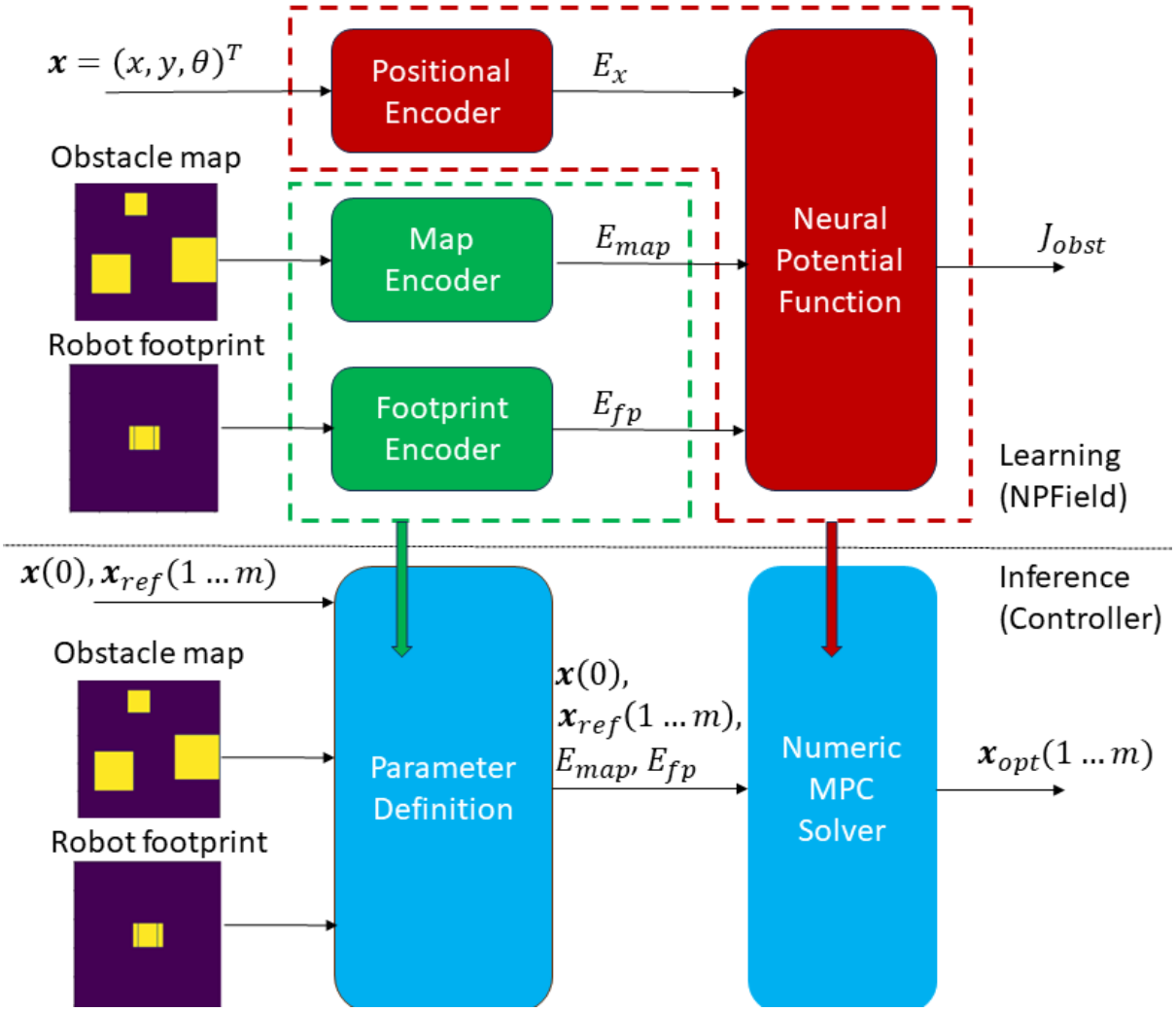


Рис 5-Общая схема предлагаемого подхода – источник [11]

Для оптимизации траектории движения роботов используют модель нейронного потенциального поля (Neural Potential Field, NPField). Она состоит из двух основных частей — нейронной сети и контроллера. В нейронной сети есть два ключевых компонента: Первый — блок энкодеров, который обрабатывает важные входные данные: положение и ориентацию робота, а также карту с препятствиями. Второй — нейронная функция потенциала (NPFunction). Она вычисляет значение отталкивающего потенциала для

конкретной позиции робота, и именно этот потенциал помогает оптимизировать путь движения.

Контроллер тоже разбит на два блока: Первый отвечает за определение параметров. На основе текущих данных он выбирает набор параметров для задачи модельно-предиктивного управления (MPC). Это происходит один раз за каждую итерацию цикла управления.

Второй — численный решатель MPC, который уже решает задачу оптимизации, используя эти параметры.Интересно, что NPField сначала обучают как единую нейросеть, а потом «разбивают» на две части: энкодеры идут в блок определения параметров, а NPFunction — интегрируется в численный решатель MPC [11]. При работе контроллера энкодеры вызываются всего один раз за цикл, а NPFunction — несколько раз, когда происходит итеративная оптимизация. Для практической реализации системы планирования траектории мобильных роботов разработали модуль численного решения MPC с использованием фреймворков Acados и CasADi. В эту систему встроена нейросетевая модель NPField, которая помогает роботу избегать препятствий.Обучение модели NPField проходило на огромном наборе данных — около 4 миллионов образцов, сгенерированных на основе городских карт MovingAI. Эти данные учитывают разные размеры и формы роботов, что делает модель универсальной и гибкой [11].Acados и CasADi используются для быстрого и эффективного решения сложных задач нелинейной оптимизации, давая разработчикам возможность сосредоточиться на математической постановке проблемы. А библиотека L4CasADi помогает плавно интегрировать нейросетевую модель в решение задачи, обеспечивая стабильную работу всей системы. В ходе экспериментов траектории, сгенерированные NPField, показали себя безопаснее и плавнее, чем у алгоритма CIAO, который часто не мог найти допустимый путь из-за жестких ограничений на избегание столкновений. Система протестирована на 20 разных сценариях и сравнивалась с другими алгоритмами — RRT, RRT*, Informed RRT и SBL [11]. Результаты показали, что NPField держится на уровне лучших планировщиков по времени работы, длине пути и плавности движения.

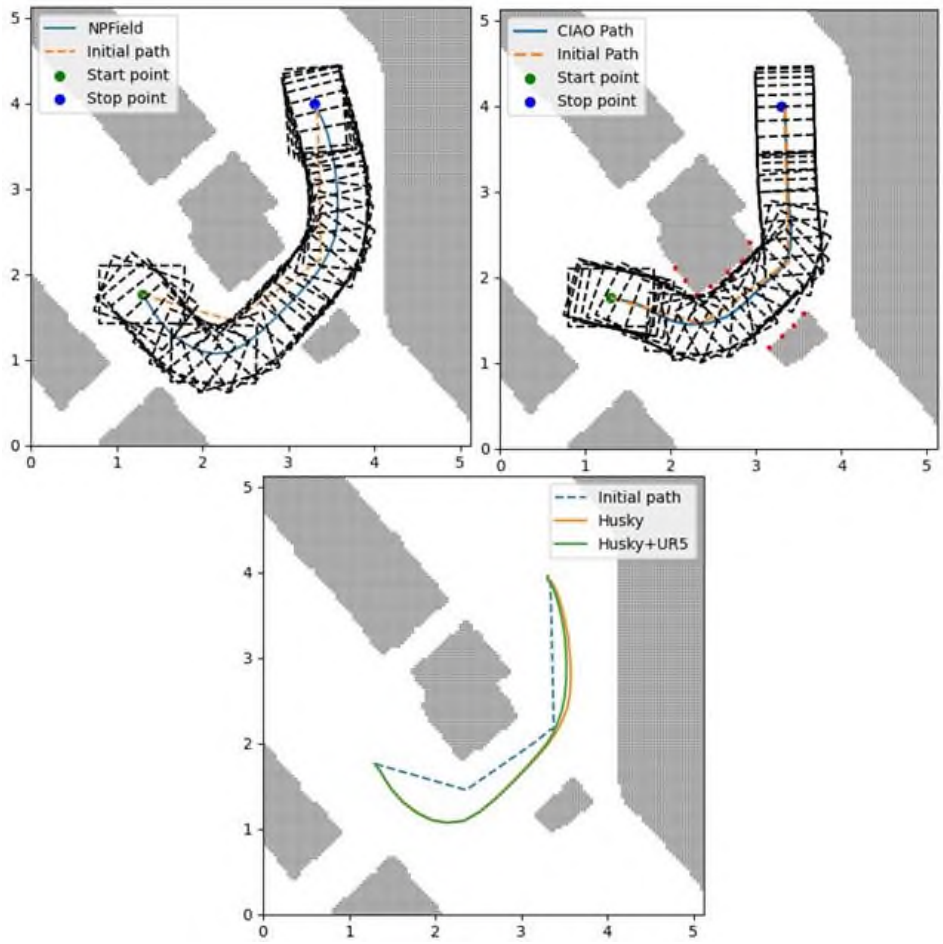


Рисунок 6- Пример сценария для локального планирования источник [11]

Слева — траектория, которую построил NPField, а справа — маршрут, предложенный другим популярным алгоритмом — CIAO. Внизу — графики траекторий NPField для разных «следов» робота [11].Эту систему проверяли на настоящем мобильном манипуляторе Husky UGV (рис. 1.2). Результаты показали, что подход действительно работает — робот уверенно и безопасно перемещался даже в сложных условиях. Это отличный пример того, как можно реализовать умное планирование пути для мобильных роботов, чтобы они могли легко и без проблем проходить через любые трудности в реальной жизни [9].

Рисунок 7- Мобильный манипулятор Husky, транспортирующий шляпу источник [11]

В итоге, разработанная система планирования траектории для мобильных роботов на базе нейросетевой модели NPField и методики модельно-предиктивного управления (MPC) показала себя с лучшей стороны — она эффективно и безопасно работает даже в сложных условиях. Благодаря интеграции с фреймворками Acados и CasADi система умеет решать сложные задачи нелинейной оптимизации в реальном времени [11]. Испытания на реальном мобильном манипуляторе Husky UGV подтвердили её преимущества по сравнению с другими алгоритмами, например CIAO — особенно в части безопасности и плавности движения. Всё это делает такую систему очень перспективной и готовой к применению в разных робототехнических задачах.

2.6 Разработка виртуальной среды для тестирования

Создание виртуальной среды для тестирования — один из самых важных этапов в оптимизации алгоритмов управления роботами. Такая среда дает возможность строить реалистичные модели роботов и их окружения, позволяя проверять алгоритмы без необходимости создавать физические прототипы [2]. Это значительно ускоряет процесс разработки и сокращает расходы на производство и обслуживание реальных моделей. Для этого используют

специальные инструменты, например CoppeliaSim, Gazebo, V-REP и другие, которые умеют точно моделировать физические процессы — гравитацию, трение, столкновения и т.п. [1]. Часто эти платформы интегрируют с системами управления, такими как ROS, чтобы симуляция максимально приближалась к реальным условиям работы роботов.

Особенность	Описание
Реализм физического моделирования	Виртуальная среда должна точно воспроизводить физические взаимодействия, такие как гравитация, трение, столкновения и другие параметры, чтобы обеспечить адекватное тестирование алгоритмов в условиях, максимально приближенных к реальным.
Гибкость настройки параметров	Возможность изменять параметры среды (например, погодные условия, освещение, время суток) и конфигурацию робота для проверки его работы в различных сценариях и условиях эксплуатации.
Моделирование взаимодействий	Имитация взаимодействия робота с окружающей средой, включая статические и динамические объекты, а также моделирование группового поведения роботов для решения комплексных задач.
Интеграция с алгоритмами управления	Виртуальная среда должна поддерживать интеграцию с алгоритмами управления, что позволяет тестировать их работу в режиме реального времени и оценивать эффективность в зависимости от заданных условий.
Поддержка обучения и анализа	Возможность использовать виртуальную среду для обучения операторов или проверки правильности работы алгоритмов, а также анализа поведения робота в нестандартных ситуациях.
Экономия ресурсов	Устранение необходимости физического прототипирования снижает затраты на разработку

Таблица 2 - Особенности и отличительные признаки создания VR

Этапы создания виртуальной среды для тестирования алгоритмов управления роботами [3] Проектирование цифровой модели — тут создают цифрового двойника робота и его окружения. Моделируют геометрию, физические свойства (масса, трение, инерция) и динамику движения. Это помогает внимательно изучить и проверить поведение робота в разных условиях [1]. Например, часто используют MATLAB и Simscape для таких моделей. Настройка физического движка — выбирают движок типа Bullet или MuJoCo, который отвечает за реалистичное моделирование столкновений, гравитации и других взаимодействий. Это делает симуляцию максимально точной и помогает

проверять автономные системы в контролируемой среде. Создание сценариев тестирования — разрабатывают разные ситуации с элементами окружения: статичными (стены) и динамическими (движущиеся двери, препятствия) [2]. Хорошо, если сценарии гибкие и расширяемые — чтобы можно было добавлять новые объекты без переделки всей модели. Для этого используют концепцию «компонуемых и исполняемых сценариев», которая облегчает управление такими тестами. Интеграция алгоритмов управления — подключают алгоритмы к виртуальной среде, чтобы проверять их работу в реальном времени. Для этого применяют ROS или другие системы. Например, описывают интеграцию алгоритмов обнаружения дефектов на базе YOLOv7 в виртуальную симуляцию. Имитация взаимодействия с пользователем — создают интерфейсы, которые позволяют человеку взаимодействовать с роботом через виртуальную среду. Часто используют VR-гарнитуры или другие интерактивные устройства [3]. Тестирование и оптимизация — проводят тесты для проверки точности работы алгоритмов, устойчивости к сбоям и общей эффективности робота в разных условиях. Исследования показывают, что симуляции помогают найти ошибки в навигационных системах роботов. Валидация результатов — сравнивают данные симуляций с реальными экспериментами (если есть возможность) или делают множество повторных симуляций, чтобы оценить надежность алгоритмов.

3. Апробирование алгоритмов в виртуальной среде
3.1 Выбор инструментов и технологий

При выборе инструментов для разработки виртуальной среды в робототехнике важно учитывать цели проекта и требования к реалистичности моделирования. Для точного моделирования физических взаимодействий подходят MuJoCo и Bullet, тогда как для простых моделей и быстрого старта удобны TinkerCAD и Virtual Robotics Toolkit. Необходимо обращать внимание на интеграцию с системами управления, например ROS, а также на технические ресурсы и стоимость лицензий. Некоторые инструменты, например TRIK Studio, имеют бесплатные версии и поддержку на русском языке. Желательно выбирать платформы, которые позволяют настраивать среду и поведение роботов для создания разнообразных сценариев. Ответственный подход к выбору инструментов гарантирует эффективное использование виртуальной среды и высокое качество разработки алгоритмов управления. В робототехнике используется множество языков и программ для создания алгоритмов управления. Программы управления движением позволяют контролировать повороты, движение вперёд и назад, а также остановку робота. Такие программы часто реализуются с использованием ROS, что позволяет тестировать алгоритмы в реальном времени. Программы обработки данных с датчиков анализируют информацию с ультразвуковых и инфракрасных датчиков для обнаружения препятствий и корректировки движения, что повышает безопасность и точность навигации. Программы навигации с картами применяют алгоритмы SLAM (Simultaneous Localization and Mapping), позволяя роботу ориентироваться в пространстве, перемещаться по маршруту и адаптироваться к изменениям окружения. Также важна программа взаимодействия с пользователем, обеспечивающая управление роботом через голосовые команды или веб-интерфейс, что делает робота более удобным в использовании. CoppeliaSim (ранее V-REP) — мощная симуляторная среда для создания реалистичных моделей роботов и их окружения, широко используемая для тестирования и оптимизации алгоритмов управления. Она поддерживает несколько физических движков, таких как MuJoCo, Bullet Physics, ODE, Vortex и Newton Dynamics, что позволяет точно моделировать столкновения и движение объектов. Среда поддерживает расчёт прямой и обратной кинематики, важный для управления сложными механизмами, например роботами-манипуляторами. В CoppeliaSim можно симулировать различные сенсоры — датчики расстояния, камеры — для проверки алгоритмов обработки данных в реалистичных условиях.

Пользовательские интерфейсы создаются с помощью плагина Qt, упрощая взаимодействие с симуляцией и отображение результатов. Среда кроссплатформенная и работает на Windows, macOS и Linux. В ней можно создавать собственные модели приводов, конструкций и других динамических объектов с визуализацией. Использование CoppeliaSim даёт несколько преимуществ: позволяет тестировать алгоритмы без физического прототипирования, снижая затраты и ускоряя разработку; обеспечивает безопасное тестирование в различных сценариях без риска повреждения оборудования; поддерживает несколько языков программирования, что облегчает интеграцию и оптимизацию. Среда широко применяется в образовательных целях и промышленности, подходит для дистанционного обучения и проведения соревнований по робототехнике.

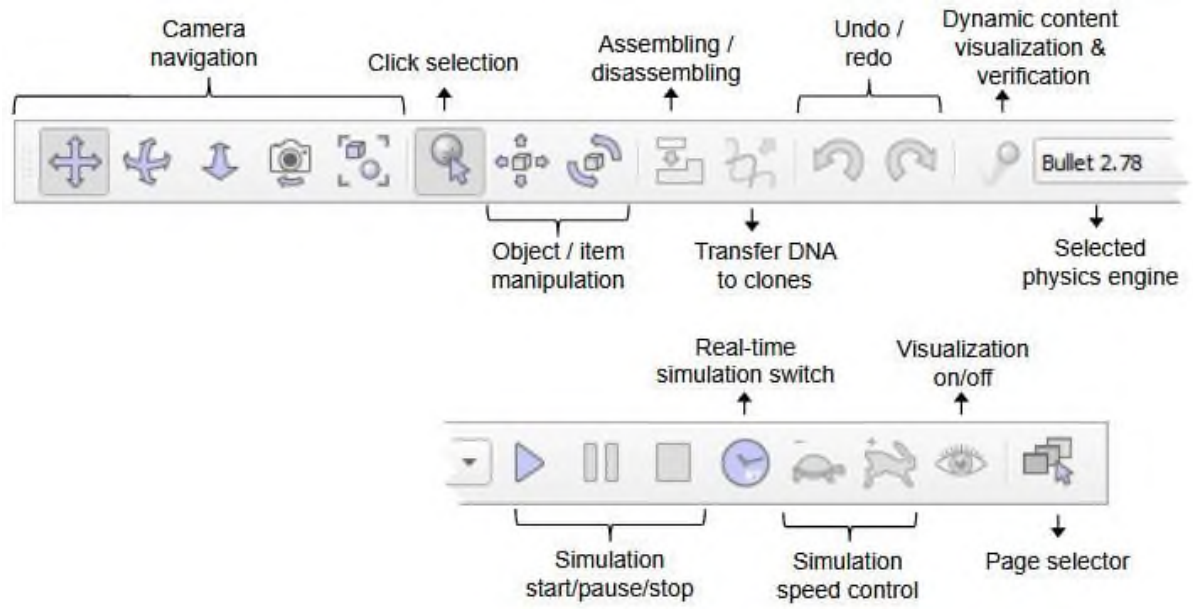


Рисунок 8-Интерфейс программы источник [14]

3.2 Создание виртуальных моделей роботов

CoppeliaSim предоставляет широкий выбор готовых моделей роботов, которые можно использовать для симуляций. Вот основные категории моделей: 1.Промышленные роботы KUKA KR16: Робот-манипулятор, подходящий для задач сборки и перемещения объектов. Используется в симуляциях "Pick-and-Place".ABB IRB 140: Компактный шестизвенный манипулятор для точных операций. Universal Robots UR5/UR10: Роботы с открытой архитектурой, популярные для обучения и исследований

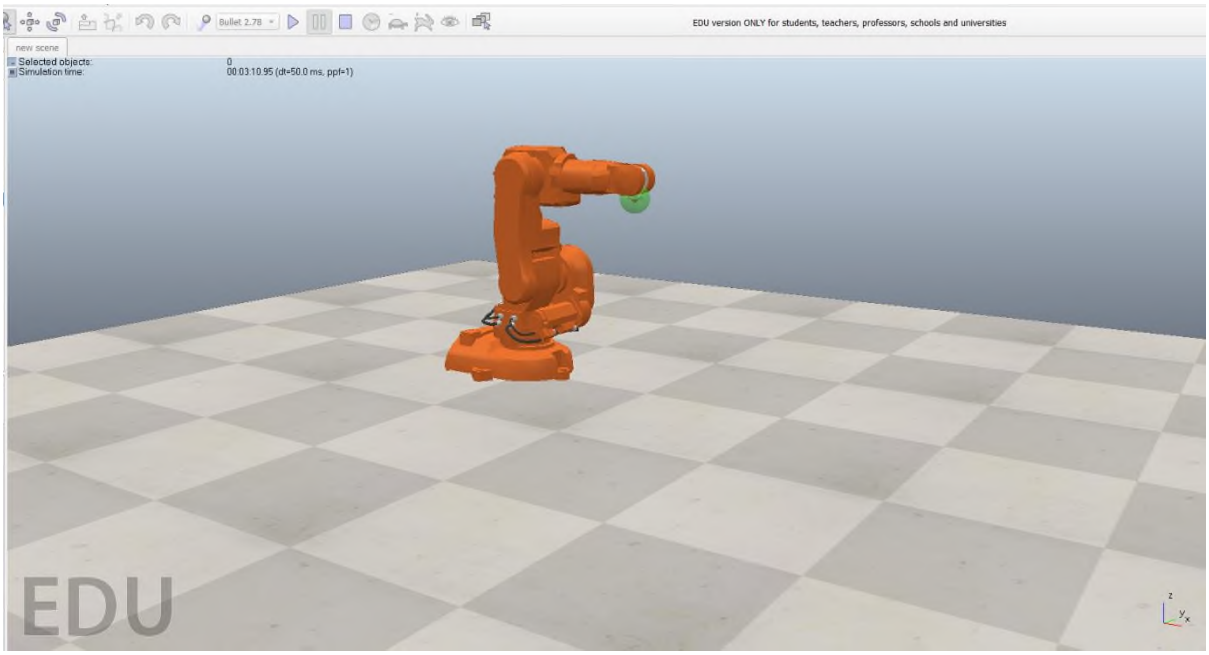


Рисунок 9-ABB IRB 140

2.Мобильные роботы Pioneer 3-DX: Двухколёсный робот с датчиками лидар и ультразвуковыми сенсорами, часто используется для задач навигации.
TurtleBot: Платформа с открытым исходным кодом, популярная для экспериментов с ROS.
Мобильные платформы Omni: Роботы с омнинаправленными колёсами для сложных траекторий движения.

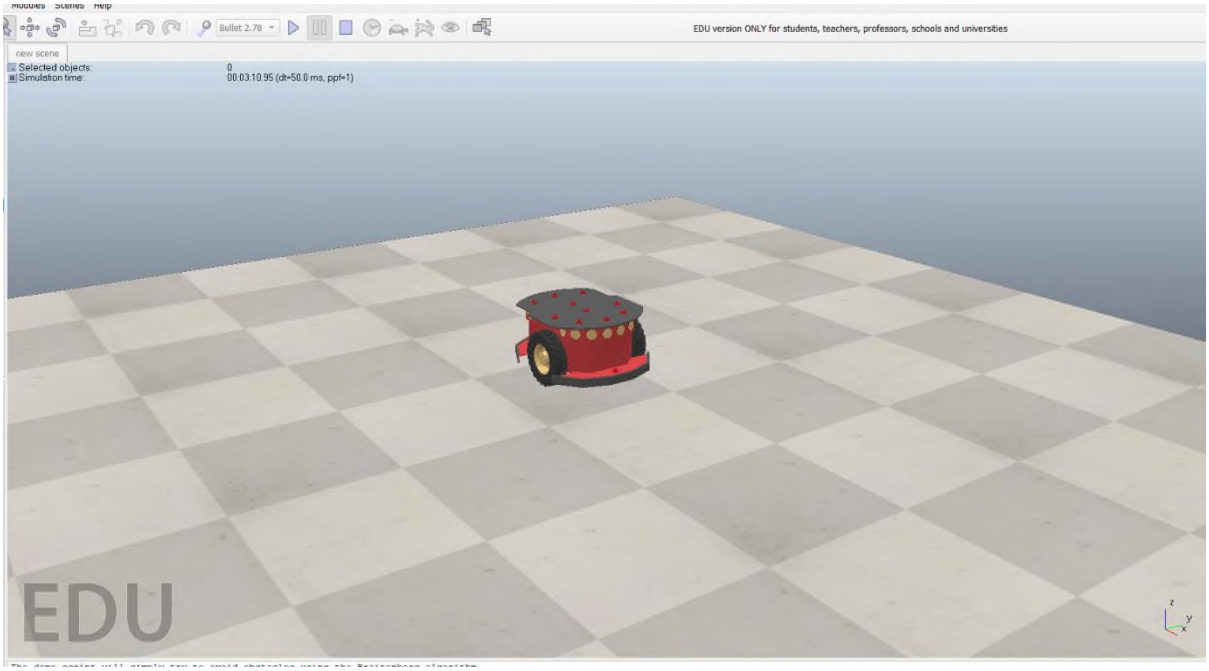


Рисунок 10- Pioneer 3-DX

3.Гуманоидные роботы NAO Robot: Гуманоид с возможностью программирования движений и поведения. Baxter Robot: Двуручный робот, используемый для симуляции взаимодействия с объектами.

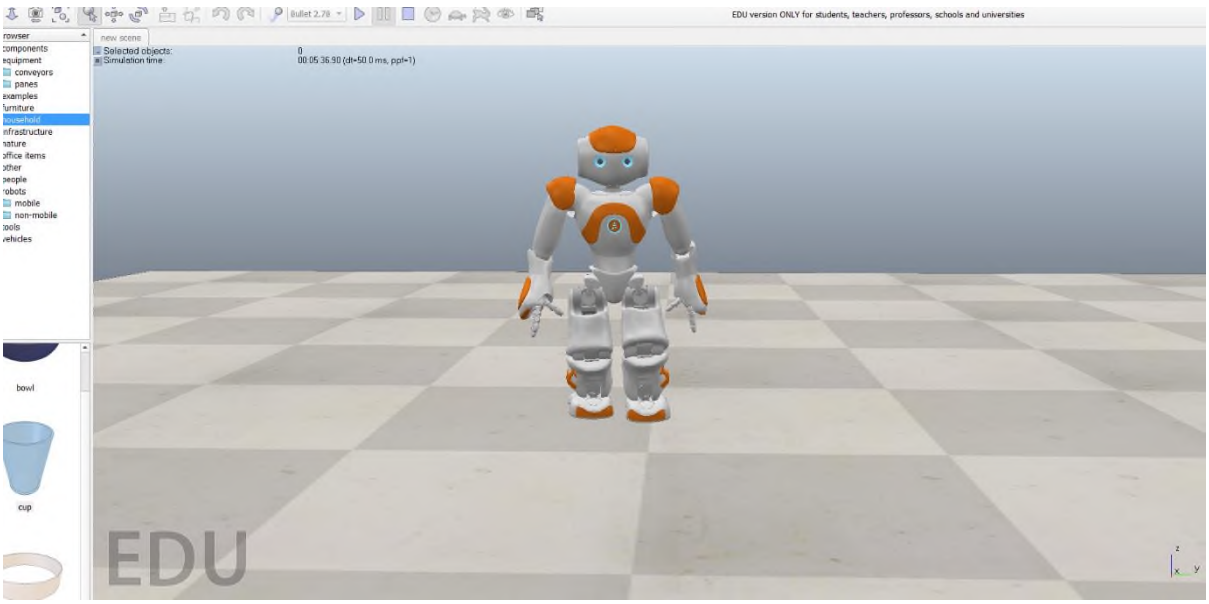


Рисунок 11-Nao Robot

4.Параллельные роботы Delta-роботы: Высокоскоростные манипуляторы для упаковки и сортировки. Stewart Platform: Платформа для симуляции движения в шести степенях свободы.

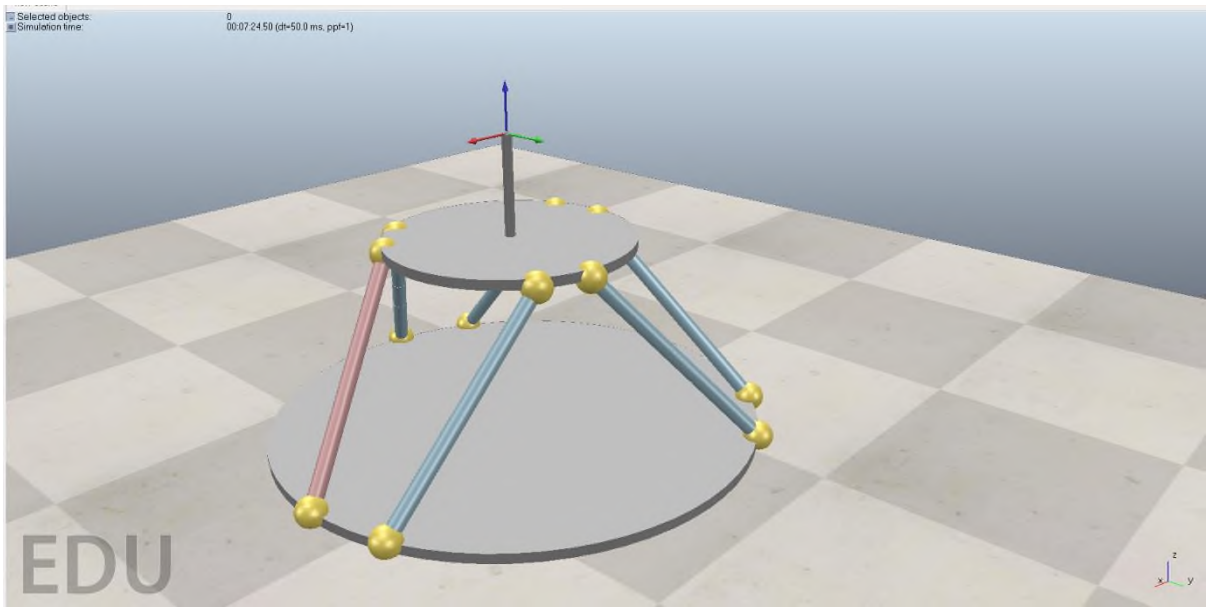


Рисунок 12-Stewart Platform

5. Другие специализированные моделиДроны (Quadcopters): Для задач воздушной навигации и мониторинга.Подводные аппараты (ROVs): Для исследования подводной среды.
Конвейеры и захваты: Используются в задачах автоматизации производственных линий.

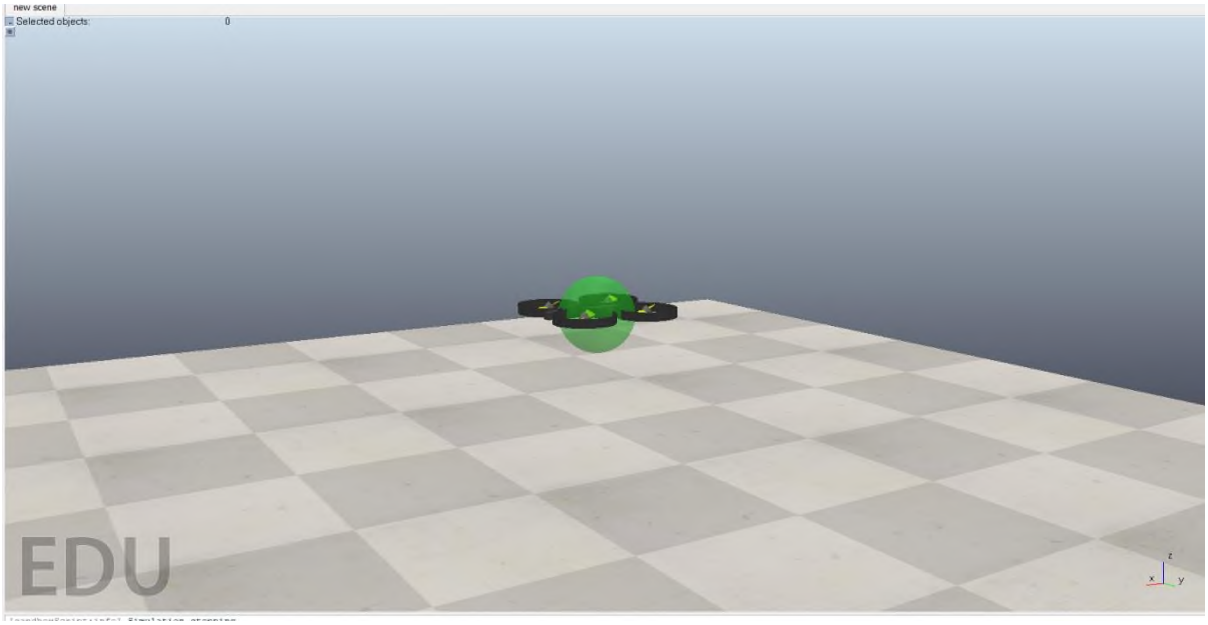


Рисунок 13 – Quadrocopter

3.3 Реализация алгоритмов управления на примере робота KUKA Yobot:

KUKA youBot — мобильный робот-манипулятор для исследований и образования.

Параметр	Значения
Производитель	Германия
Тип	Мобильный робот манипулятор с 5 степенями свободы
Нагрузка	10 кг манипулятор,20 кг платформа
Рабочий радиус	901 мм
Повторяемость	0,03 мм
Скорость платформы	До 0.8 м/с
Энергосистема	Свинцово кислотные батареи(12 В,5 А*ч)время работы около 90 минут
Програмнное обеспечение	Открытый программный код

Таблица 3 - Основные характеристики робота

Для данного примера я буду использовать программирование на языке Lua, встроенном в саму программу. Lua — легковесный, гибкий язык для скриптов и встраивания. Он довольно прост и удобен, и для данного примера мне отлично подойдёт.

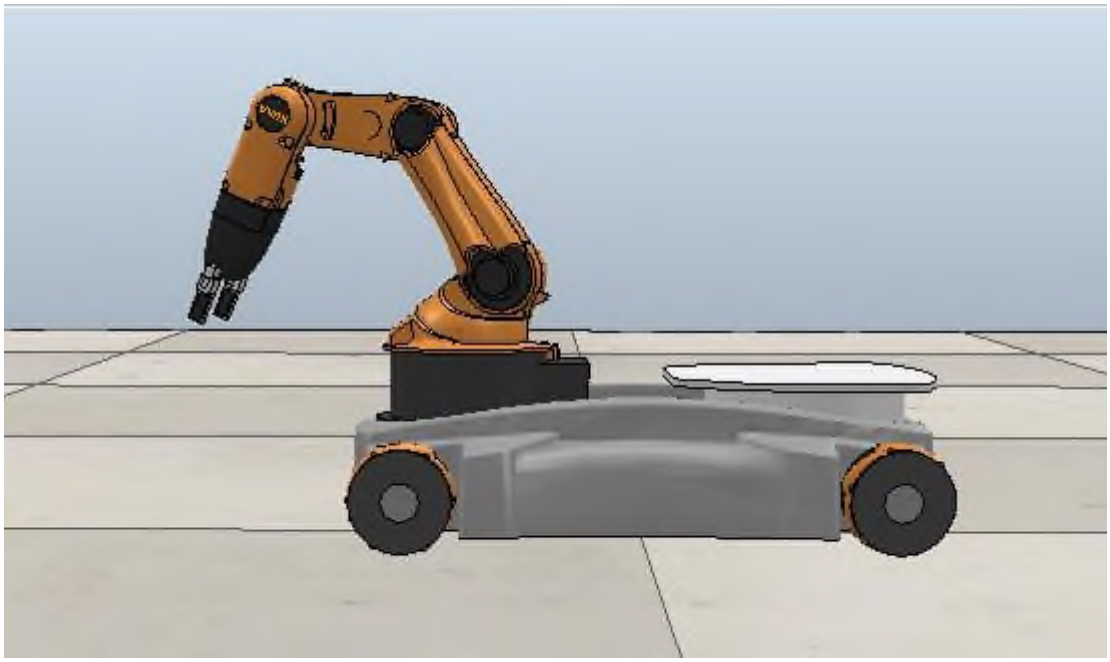


Рисунок 14 - Вид сбоку YouBot

Рисунок 15 - Вид спереди YouBot

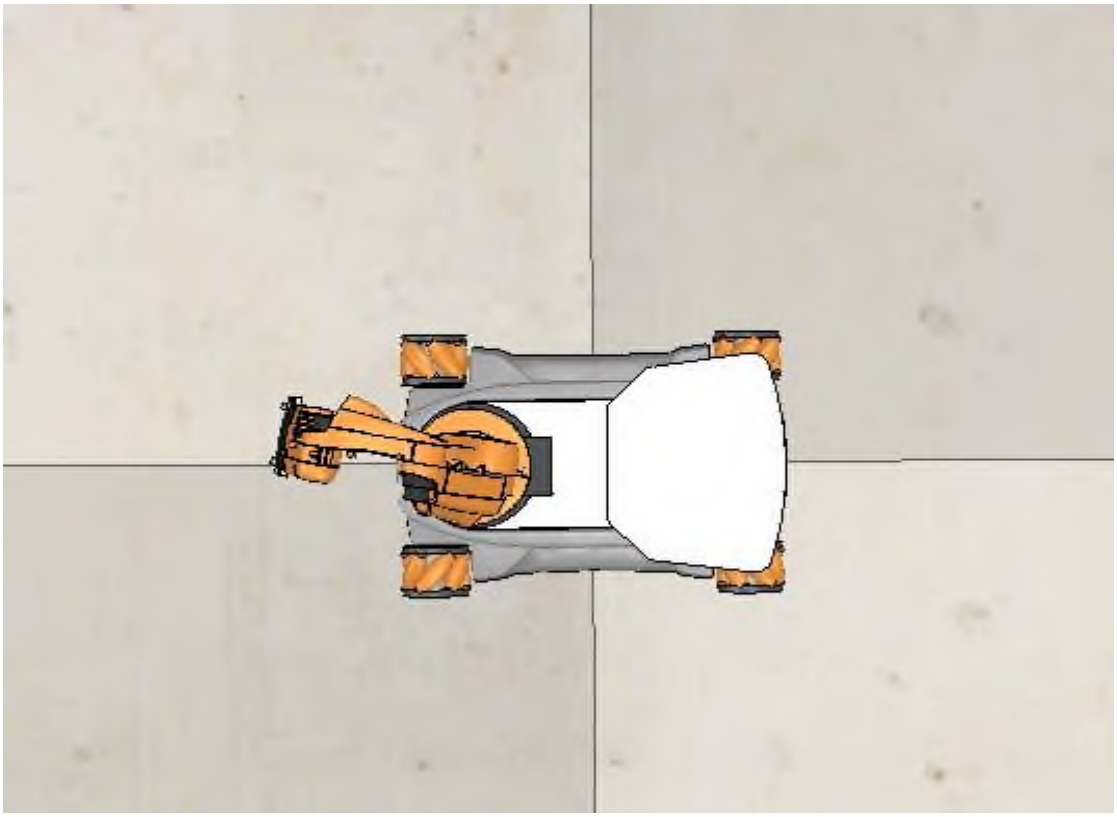
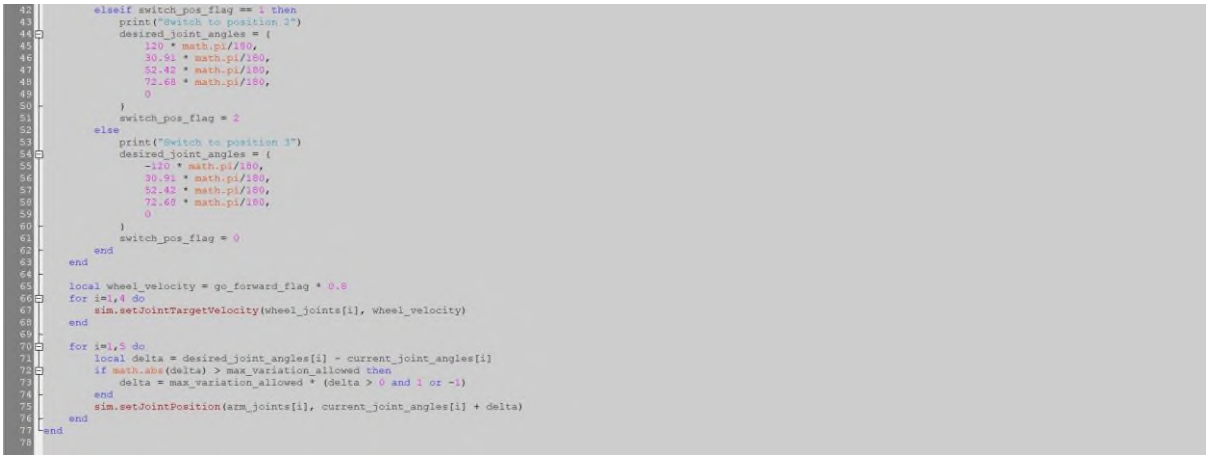


Рисунок 16 - Вид сверху

В данном случае я постарался реализовать простые алгоритмы управления роботом, что видно на скриншоте листинга кода (Рисунки 17-18).

```
function sysCall_init()
    you_bot = sim.getObjectHandle('youBot')
    wheel_joints = {
        sim.getObjectHandle('rollingJoint_fl'),
        sim.getObjectHandle('rollingJoint_rl'),
        sim.getObjectHandle('rollingJoint_rr'),
        sim.getObjectHandle('rollingJoint_fr')
    }
    arm_joints = {}
    for i=0,4 do
        arm_joints[i+1] = sim.getObjectHandle('youBotArmJoint'..i)
    end
    desired_joint_angles = {0, 0, 0, 0, 0}
    for i=1,5 do
        sim.setJointPosition(arm_joints[i], desired_joint_angles[i])
    end
    max_joint_velocity = 50 * math.pi / 180
    switch_pos_flag = 0
    go_forward_flag = 1
    last_switch_time = sim.getSimulationTime()
end

function sysCall_actuation()
    current_joint_angles = {}
    for i=1,5 do
        current_joint_angles[i] = sim.getJointPosition(arm_joints[i])
    end
    local simu_time = sim.getSimulationTime()
    local max_variation_allowed = max_joint_velocity * sim.getSimulationTimeStep()

    if simu_time % 10 < 0.05 then
        print('\nSimulation time: '..string.format("%.1f", simu_time))
        if simu_time - last_switch_time >= 60 then
            go_forward_flag = -go_forward_flag
            last_switch_time = simu_time
        end

        if switch_pos_flag == 0 then
            print("Switch to position 1")
            desired_joint_angles = {math.pi, 0, 0, 0, 0}
            switch_pos_flag = 1
        elseif switch_pos_flag == 1 then
            print("Switch to position 2")
            desired_joint_angles = {
                120 * math.pi/180,
                30.91 * math.pi/180,
                52.42 * math.pi/180,
                72.68 * math.pi/180,
                0
            }
            switch_pos_flag = 2
```

Рисунок 17 - Листинг кода 1 часть



```
        elseif switch_pos_flag == 1 then
            print("Switch to position 2")
            desired_joint_angles = {
                120 * math.pi/180,
                30.91 * math.pi/180,
                52.42 * math.pi/180,
                72.68 * math.pi/180,
                0
            }
            switch_pos_flag = 2
        else
            print("Switch to position 3")
            desired_joint_angles = {
                -120 * math.pi/180,
                30.91 * math.pi/180,
                52.42 * math.pi/180,
                72.68 * math.pi/180,
                0
            }
            switch_pos_flag = 0
        end
    end

    local wheel_velocity = go_forward_flag * 0.8
    for i=1,4 do
        sim.setJointTargetVelocity(wheel_joints[i], wheel_velocity)
    end

    for i=1,5 do
        local delta = desired_joint_angles[i] - current_joint_angles[i]
        if math.abs(delta) > max_variation_allowed then
            delta = max_variation_allowed * (delta > 0 and 1 or -1)
        end
        sim.setJointPosition(arm_joints[i], current_joint_angles[i] + delta)
    end
end
```

Рисунок 18 - Листинг кода часть 2

Общая концепция кода В первую очередь я провел инизицализацию с помощью команды функции sysCall_init()
Что происходит:
Получение хендлов колёс и суставов робота.
Установка начальных углов манипулятора.
Инициализация параметров управления.
Далее Основная логика функция (sysCall_actuation())
Ход работы:
Чтение текущих углов суставов.
Смена позиций манипулятора каждые 10 секунд(параметр можно менять по усмотрению).
Смена направления движения каждые 60 секунд(также можно менять).
Управление колёсами (прямолинейное движение).
Плавное движение суставов с ограничением скорости.

Алгоритмы управления: 1. Циклическая смена позиции манипулятора

```
if simu_time % 10 < 0.05 then
    if switch_pos_flag == 0 then
      desired_joint_angles = {math.pi, 0, 0, 0, 0}
      switch_pos_flag = 1 -- ... остальные условия

End
```

2. Смена направления движения

```
if simu_time - last_switch_time >= 60 then
   go_forward_flag = -go_forward_flag
   last_switch_time = simu_time
end
```

3. Плавное движение суставов

```
for i=1,5 do
    local delta = desired_joint_angles[i] - current_joint_angles[i]
    if math.abs(delta) > max_variation_allowed then
        delta = max_variation_allowed * (delta > 0 and 1 or -1)

end
    sim.setJointPosition(arm_joints[i], current_joint_angles[i] + delta)

End
```

Изображения работы робота и алгоритмов скрипта :

Рисунок 18-Первое положение работы алгоритма

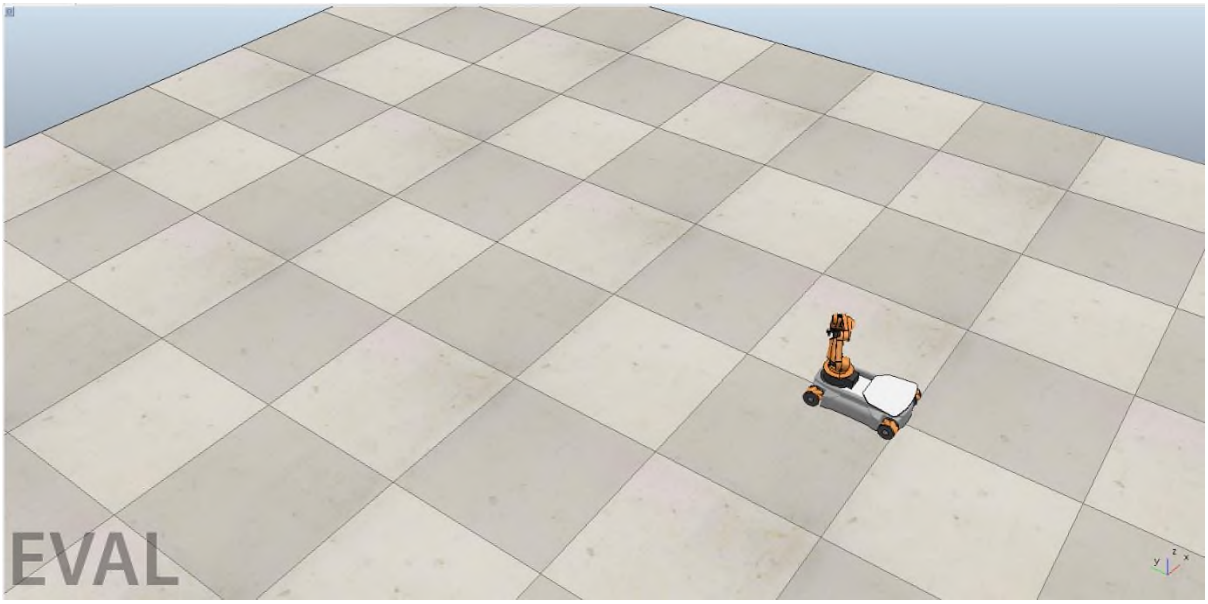


Рисунок 19-Первое положение работы алгоритма

Рисунок 20 - Второе положение работы алгоритма

В ходе апробирования простых алгоритмов управления я выяснил, что VR-среда отлично подходит для данной задачи. Поскольку она является очень гибкой и великолепно контролируемой, это позволяет эффективно тестировать и оттачивать различные подходы к управлению робототехническими системами. Кроме того, использование VR-среды способствует более глубокому анализу поведения алгоритмов и облегчает поиск оптимальных решений.

3.4 Реализация алгоритмов управления “Возьми и поставь” и тестирование

Проект №2: Основной проект и реализация генерируемого алгоритма управления**:** За основу также был взят робот KUKA Youbot . Он был выбран на тех же критериях, что и в опыте, рассмотренном ранее. Для данного проекта был использован язык программирования Python.

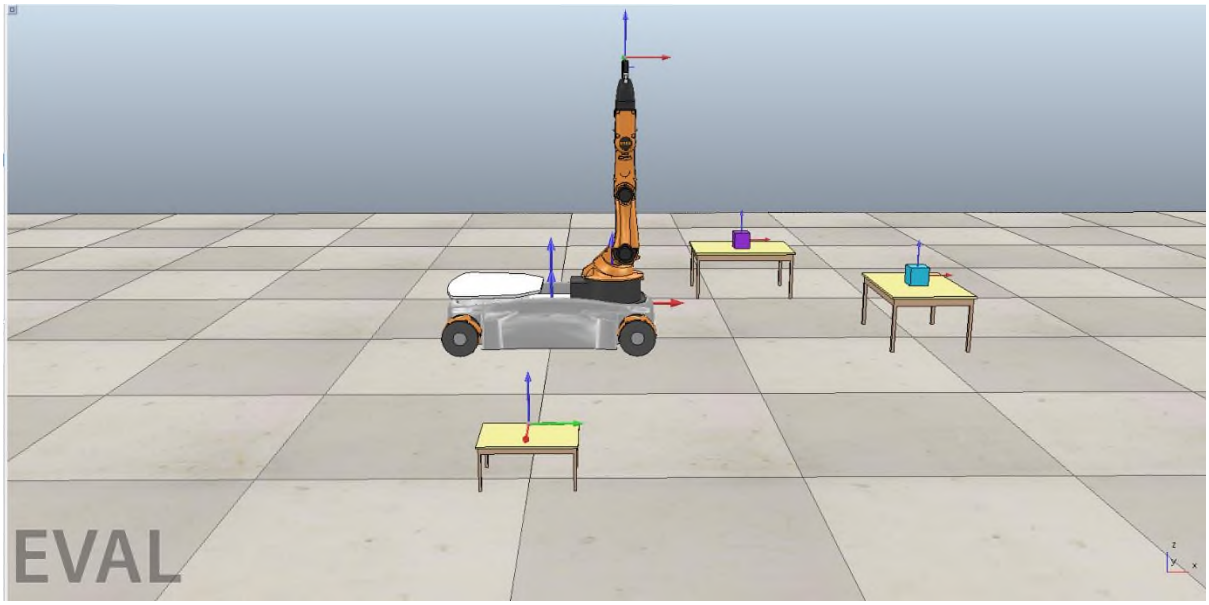


Рисунок 21 - Вид робота

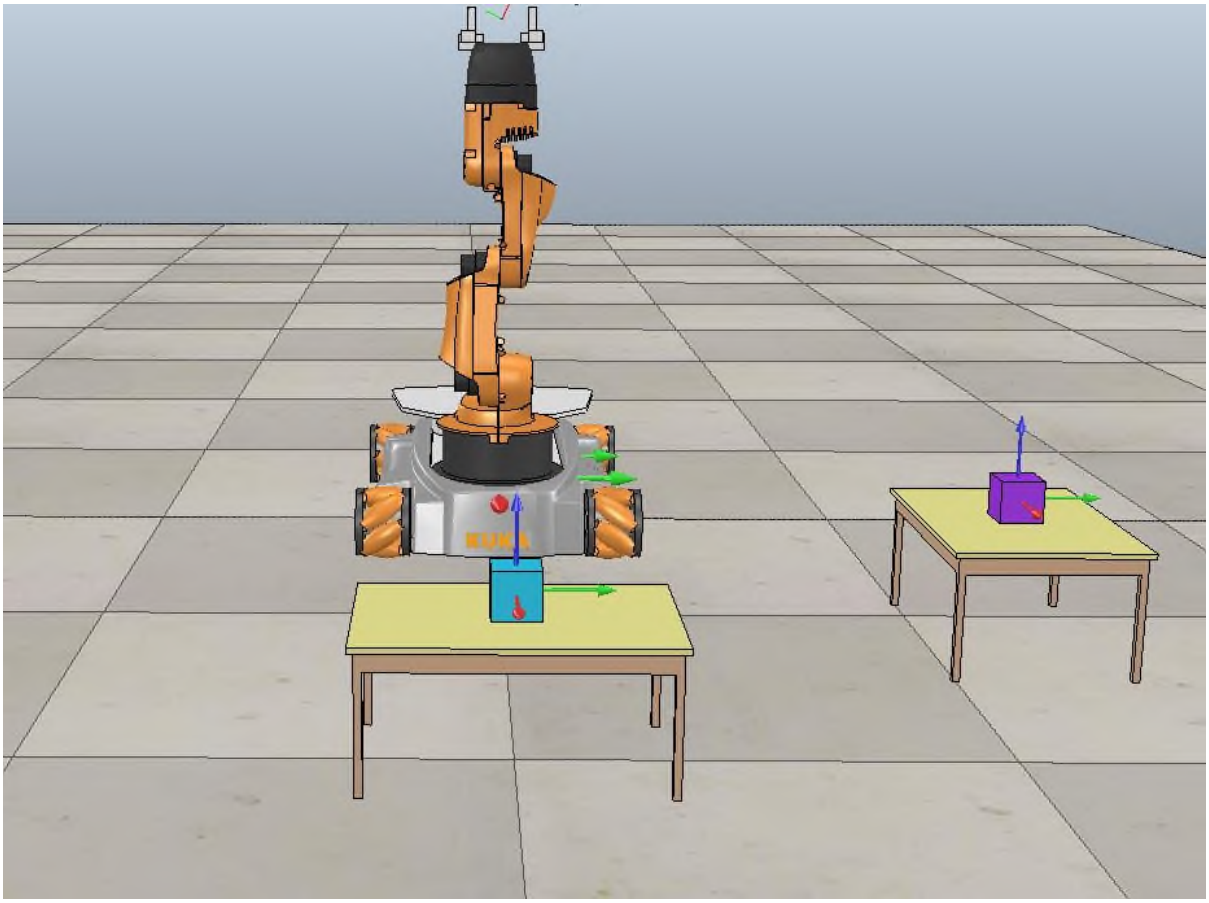


Рисунок 22 - Вид робота спереди

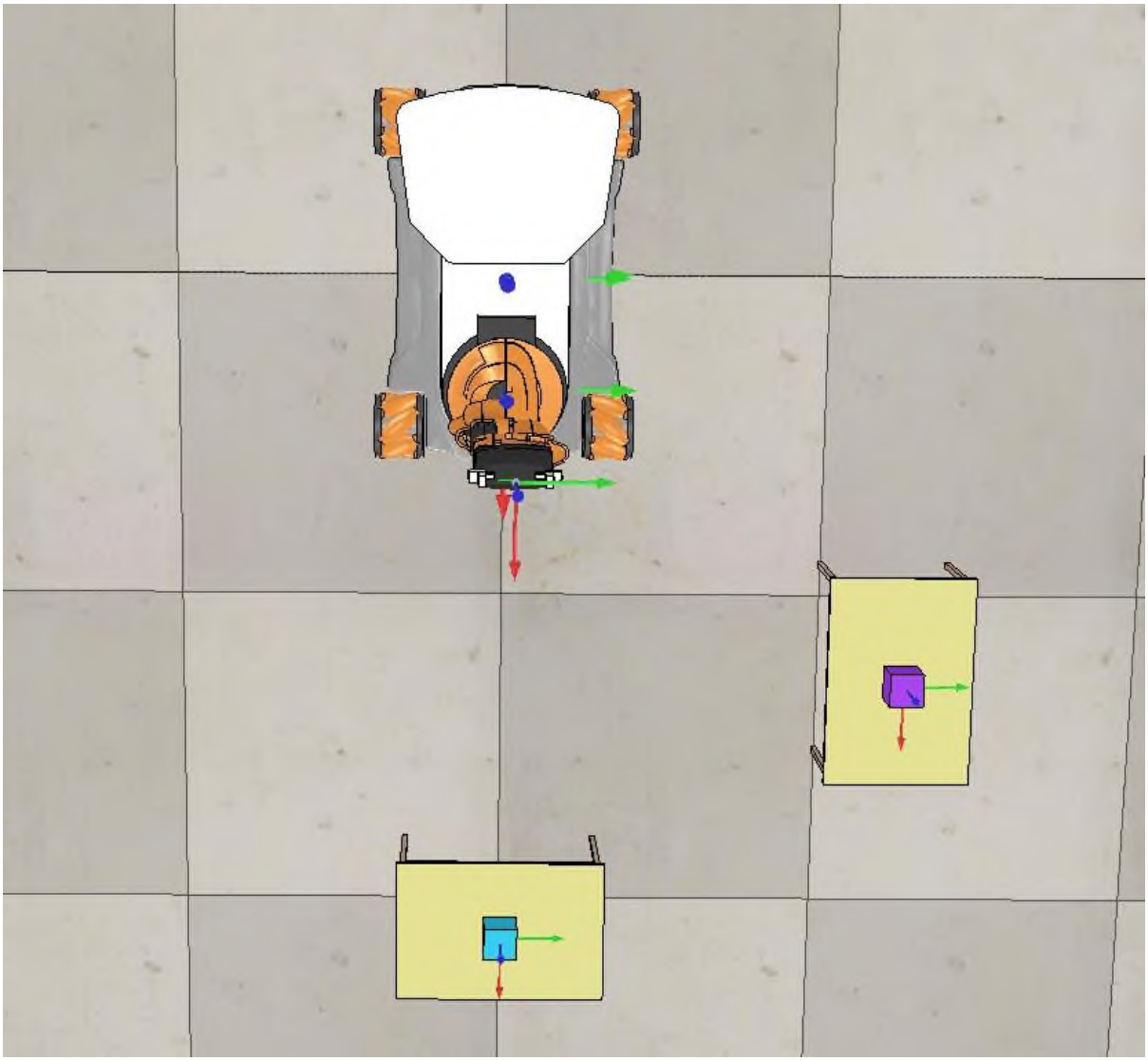


Рисунок 23-Вид робота сверху

Для обеспечения корректной работы робота я применил генерацию траектории, которая была обозначена как TrajectoryGenerator. Для правильной работы было сформировано 13 переменных, которые также отображены на рисунке 24.

№	1	2	3	4	5	6	7	8	9	10	11	12	13
Значение	phi	x	y	J1	J2	J3	J4	J5	w1	w2	w3	w4	0-1

Таблица 4 - Переменные TrajectoryGenerator

Где:x, y, φ: координаты и угол поворота суставов World_X_Joint, World_Y_Joint, World_Th_Joint
J1–J5: углы сочленений манипулятора youBot (в радианах)
w1–w4: углы поворота колёс (в радианах):
w1: переднее левое
w2: переднее правое
w3: заднее правое
w4: заднее левое
Состояние захвата:
0 — открыт
1 — закрыт

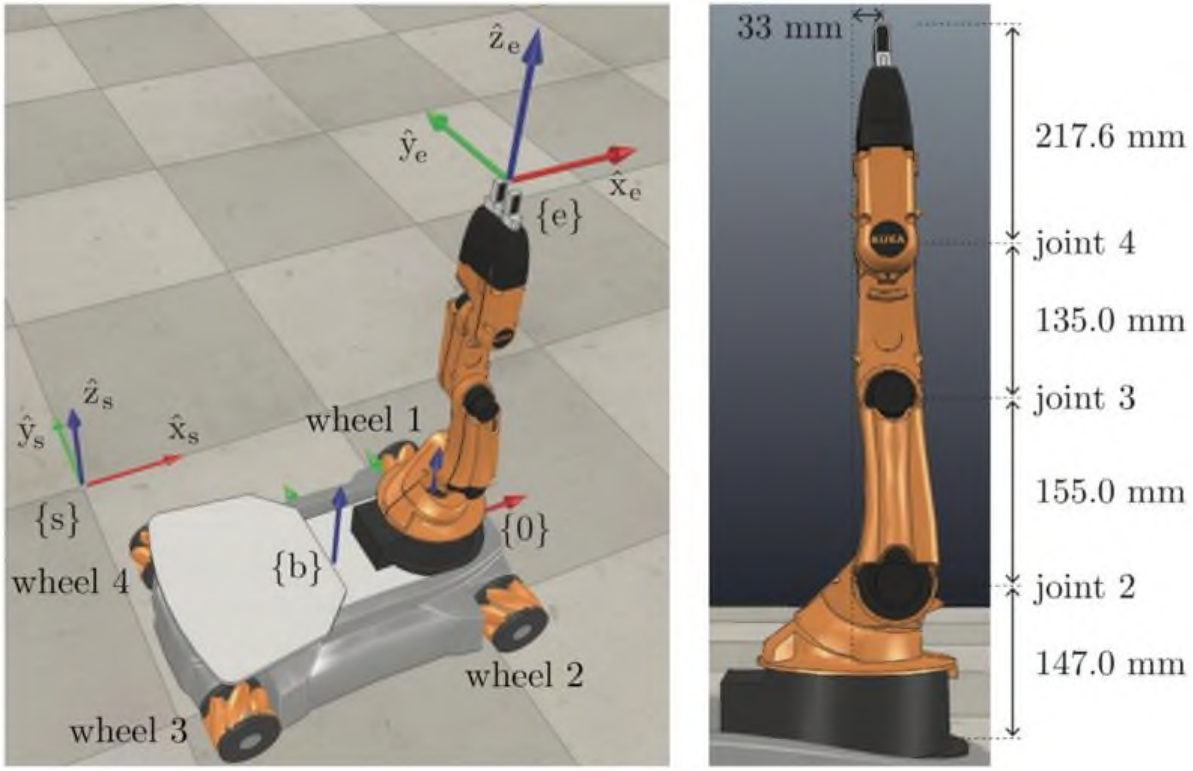


Рисунок 24-Условнные обозначения и размеры манипулятора

Также критически важны габаритные параметры для захвата (Рис. 25), они важны для точного позиционирования и избегания коллизий. Ограничения захвата: Минимальное расстояние: $d1,min=2\,см$ $d1$,min= 2см, максимальное расстояние: $d1,max=7\,см$ $d1$,max=7см, дополнительные расстояния: $d2=3.5\,см$ $d2$=3.5см, $d3=4.3\,см$ $d3$=4.3см. Геометрия платформы: Расстояние между передними и задними колёсами: $0.47\,м$, между левыми и правыми колёсами: $0.3\,м$, радиус колёс: 0.0475м.

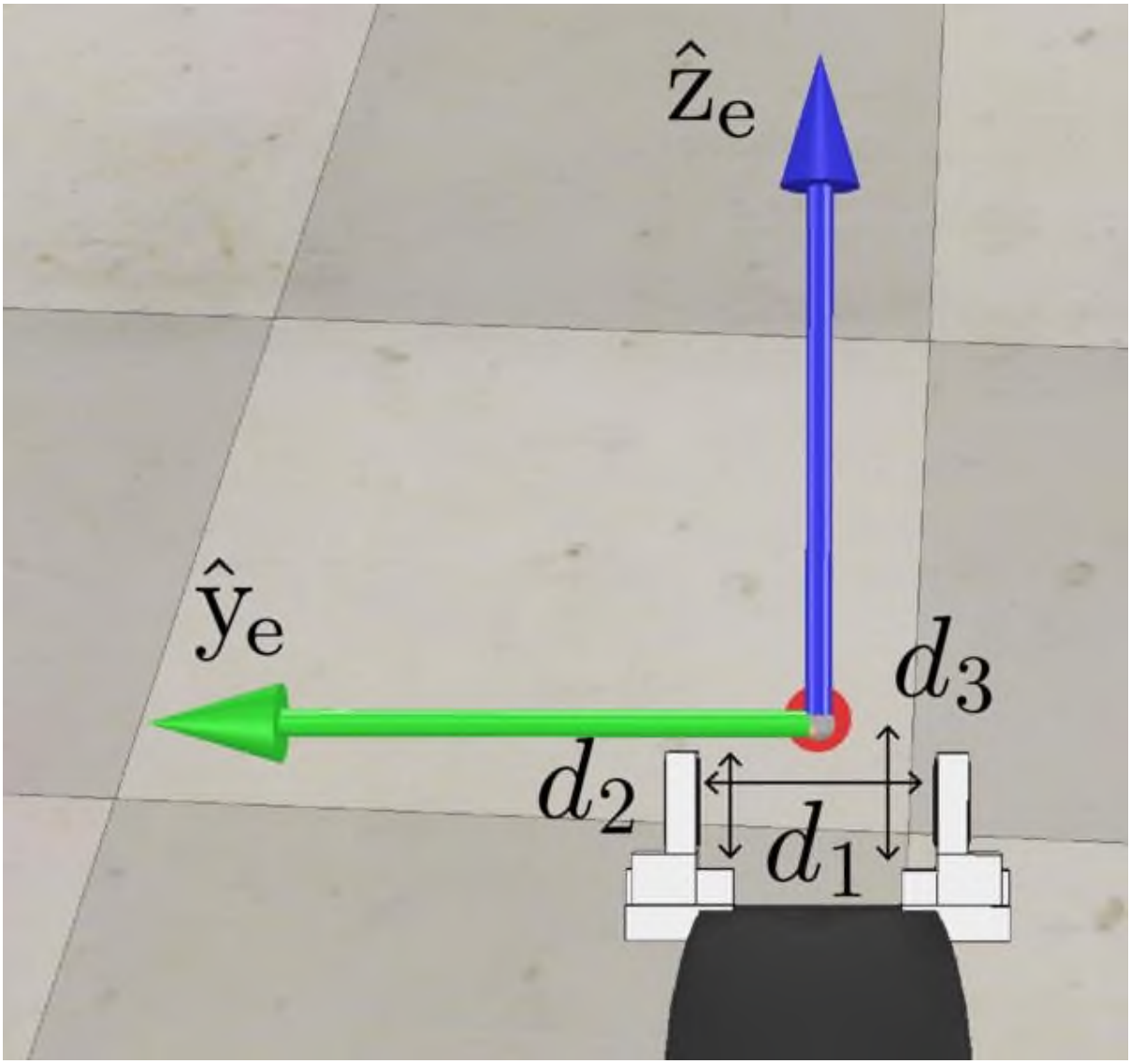


Рисунок 25-Габаритные размеры захвата

Код Trajectory_generator(Рис. 26):

```python
import numpy as np
import modern_robotics as mr

def trajectory_generator(T_se_initial, T_sc_initial, T_sc_final, T_ce_grasp, T_ce_standoff, k):
    # Сегмент 1: Перемещение в standby-позицию над объектом
    T_se_standoff_initial = T_sc_initial @ T_ce_standoff
    T_se_segment1 = mr.CartesianTrajectory(T_se_initial, T_se_standoff_initial, 5, 500, 3)

    # Сегмент 2: Спуск к объекту
    T_se_grasp = T_sc_initial @ T_ce_grasp
    T_se_segment2 = mr.CartesianTrajectory(T_se_segment1[-1], T_se_grasp, 2, 200, 3)
    T_se_before = np.append(T_se_segment1, T_se_segment2, axis=0)

    # Сегмент 3: Закрытие захвата (пауза 0.64 сек)
    T_se_before = np.append(T_se_before, np.tile(T_se_before[-1], (64, 1)), axis=0)

    # Сегмент 4: Возврат в standby-позицию
    T_se_segment4 = mr.CartesianTrajectory(T_se_grasp, T_se_standoff_initial, 2, 200, 3)
    T_se_before = np.append(T_se_before, T_se_segment4, axis=0)

    # Сегмент 5: Перемещение к целевой standby-позиции
    T_se_standoff_final = T_sc_final @ T_ce_standoff
    T_se_segment5 = mr.CartesianTrajectory(T_se_standoff_initial, T_se_standoff_final, 8, 800, 3)
    T_se_before = np.append(T_se_before, T_se_segment5, axis=0)

    # Сегмент 6: Спуск в конечную позицию
    T_se_lose = T_sc_final @ T_ce_grasp
    T_se_segment6 = mr.CartesianTrajectory(T_se_standoff_final, T_se_lose, 2, 200, 3)
    T_se_before = np.append(T_se_before, T_se_segment6, axis=0)

    # Сегмент 7: Открытие захвата (пауза 0.64 сек)
    T_se_before = np.append(T_se_before, np.tile(T_se_before[-1], (64, 1)), axis=0)

    # Сегмент 8: Возврат в standby-позицию
```

```python
def trajectory_generator(T_se_initial, T_sc_initial, T_sc_final, T_ce_grasp, T_ce_standoff, k):
    T_se_segment5 = mr.CartesianTrajectory(T_se_standoff_initial, T_se_standoff_final, 8, 800, 3)
    T_se_before = np.append(T_se_before, T_se_segment5, axis=0)

    # Сегмент 6: Спуск в конечную позицию
    T_se_lose = T_sc_final @ T_ce_grasp
    T_se_segment6 = mr.CartesianTrajectory(T_se_standoff_final, T_se_lose, 2, 200, 3)
    T_se_before = np.append(T_se_before, T_se_segment6, axis=0)

    # Сегмент 7: Открытие захвата (пауза 0.64 сек)
    T_se_before = np.append(T_se_before, np.tile(T_se_before[-1], (64, 1)), axis=0)

    # Сегмент 8: Возврат в standby-позицию
    T_se_segment8 = mr.CartesianTrajectory(T_se_before[-1], T_se_standoff_final, 2, 200, 3)
    T_se_before = np.append(T_se_before, T_se_segment8, axis=0)

    # Формирование выходной матрицы
    total_steps = int(k * 21 / 0.01 + 64 * 2)
    T_se_post = np.zeros((total_steps, 13))

    for i in range(total_steps):
        T_se_post[i, :3] = T_se_before[i, :3, 3]  # Позиция (x, y, z)
        T_se_post[i, 3:6] = T_se_before[i, 0, :3]  # Ориентация (r, p, y)
        T_se_post[i, 6:9] = T_se_before[i, 1, :3]  # Ориентация (r, p, y)
        T_se_post[i, 9:12] = T_se_before[i, 2, :3]  # Ориентация (r, p, y)
        T_se_post[i, 12] = 0

    # Активация захвата на этапах удержания объекта
    grasp_start = int(k * 7 / 0.01)
    grasp_end = int(k * 19 / 0.01 + 64)
    T_se_post[grasp_start:grasp_end, 12] = 1

    return T_se_post
```

Рисунок 26 - Листинг кода Trajectory_generator

Код состоит из 8 этапов (segments), которые формируют полный цикл работы робота. Каждый этап отвечает за конкретное действие:

1. Подход к объекту (standoff-позиция).
2. Спуск к объекту.
3. Закрытие захвата (пауза).
4. Возврат в standby-позицию.
5. Перемещение к целевой точке.
6. Спуск для размещения объекта.
7. Открытие захвата (пауза).
8. Возврат в standby-позицию.

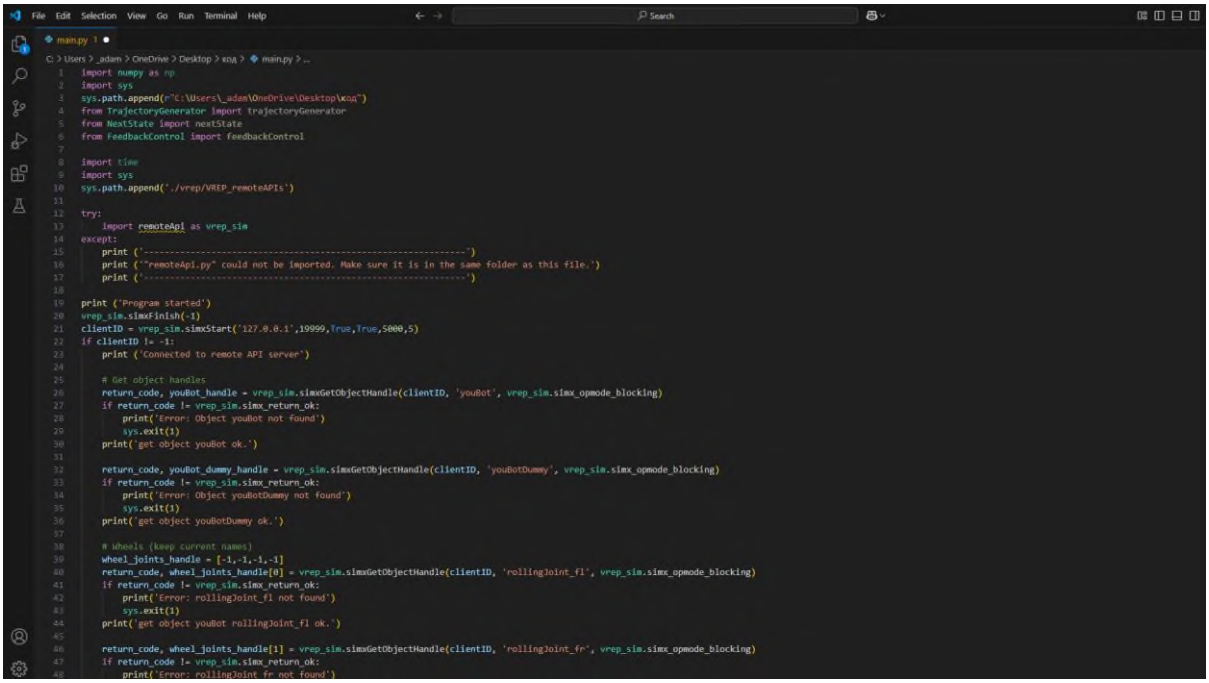
Как это работает: Первое это подготовка матриц преобразования T_sc_initial- Матрица преобразования объекта в начальной позиции. T_ce_grasp/T_ce_standoff: Матрицы смещения захвата относительно объекта (для спуска и возврата).Далее генерация траектории с помощью функции из библиотеки modern_robotics, создающая плавную траекторию между двумя точками.

T_se_segment1 = mr.CartesianTrajectory(T_se_initial, T_se_standoff_initial, 5, 500, 3)

Далее просиходит объеденение сегметнтов через команду np.append,которая объеденяет все в единный массив.Также не мало важным является формированние выходной матрицы как раз состоящей из 13 переменных обозначенных ранее

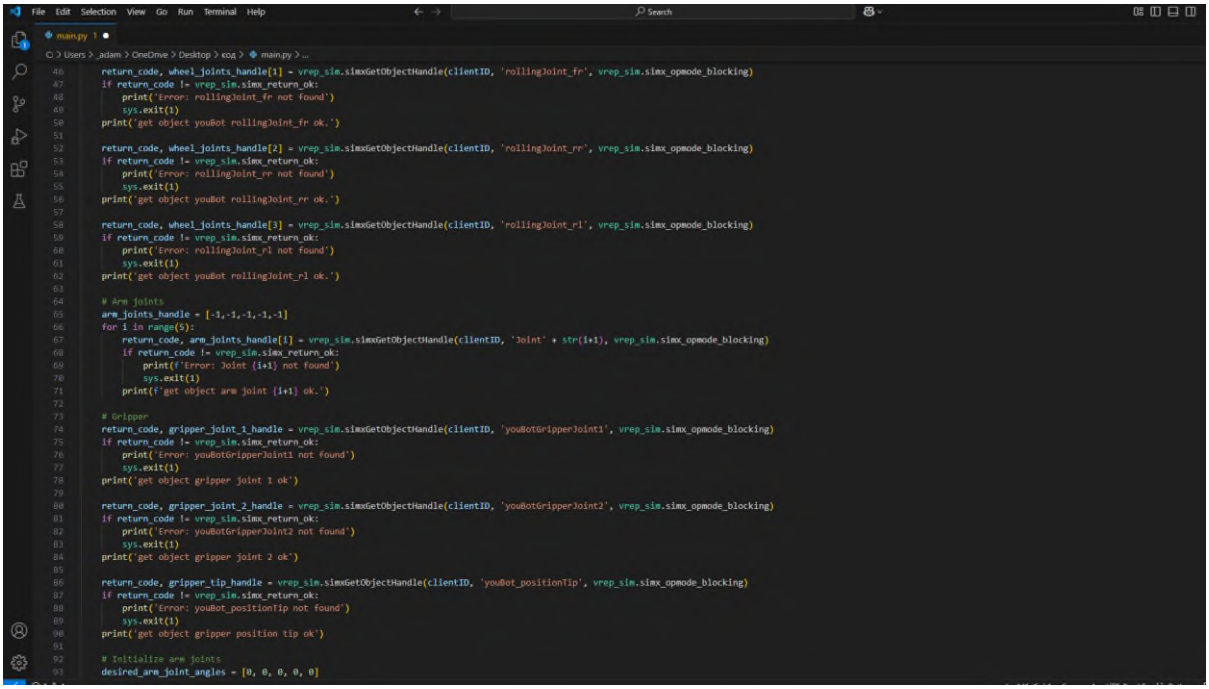
```
T_se_post[i, :3] = T_se_before[i, :3, 3] # Позиция (x, y, z)
T_se_post[i, 3:6] = T_se_before[i, 0, :3] # Ориентация (r, p, y)
T_se_post[i, 6:9] = T_se_before[i, 1, :3] # Ориентация (r, p, y)
T_se_post[i, 9:12] = T_se_before[i, 2, :3] # Ориентация (r, p, y)
T_se_post[i, 12] = 0 # Состояние захвата (по умолчанию — открыт)
```

Основной код (Рис. 27-29) в целом использует стандартные библиотеки Python (numpy, sys, time), а также специализированную библиотеку remoteAPI и пользовательские модули для реализации алгоритмов управления.[6]



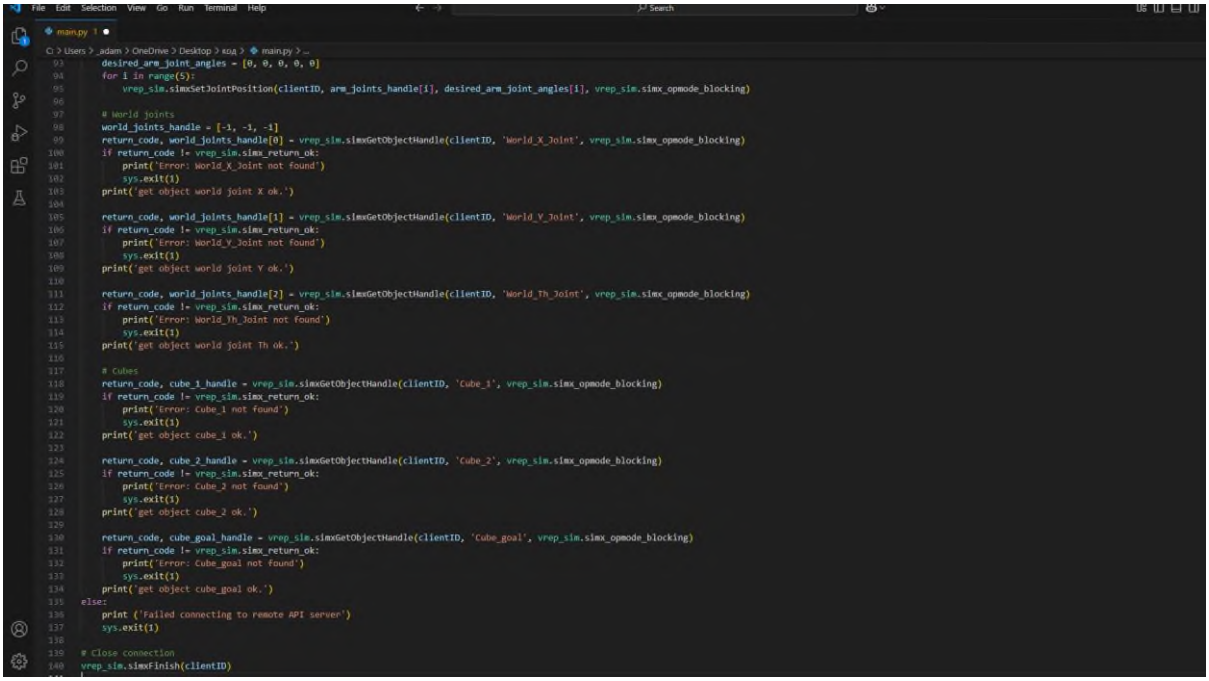
```python
import numpy as np
import sys
sys.path.append(r"C:\Users\_adam\OneDrive\Desktop\код")
from TrajectoryGenerator import trajectoryGenerator
from NextState import nextState
from FeedbackControl import feedbackControl

import time
import sys
sys.path.append('./vrep/VREP_remoteAPIs')

try:
    import remoteApi as vrep_sim
except:
    print ('--------------------------------------------------------------')
    print ('"remoteApi.py" could not be imported. Make sure it is in the same folder as this file.')
    print ('--------------------------------------------------------------')

print ('Program started')
vrep_sim.simxFinish(-1)
clientID = vrep_sim.simxStart('127.0.0.1',19999,True,True,5000,5)
if clientID != -1:
    print ('Connected to remote API server')

    # Get object handles
    return_code, youBot_handle = vrep_sim.simxGetObjectHandle(clientID, 'youBot', vrep_sim.simx_opmode_blocking)
    if return_code != vrep_sim.simx_return_ok:
        print('Error: Object youBot not found')
        sys.exit(1)
    print('get object youBot ok.')

    return_code, youBot_dummy_handle = vrep_sim.simxGetObjectHandle(clientID, 'youBotDummy', vrep_sim.simx_opmode_blocking)
    if return_code != vrep_sim.simx_return_ok:
        print('Error: Object youBotDummy not found')
        sys.exit(1)
    print('get object youBotDummy ok.')

    # Wheels (keep current names)
    wheel_joints_handle = [-1,-1,-1,-1]
    return_code, wheel_joints_handle[0] = vrep_sim.simxGetObjectHandle(clientID, 'rollingJoint_fl', vrep_sim.simx_opmode_blocking)
    if return_code != vrep_sim.simx_return_ok:
        print('Error: rollingJoint_fl not found')
        sys.exit(1)
    print('get object youBot rollingJoint_fl ok.')

    return_code, wheel_joints_handle[1] = vrep_sim.simxGetObjectHandle(clientID, 'rollingJoint_fr', vrep_sim.simx_opmode_blocking)
    if return_code != vrep_sim.simx_return_ok:
        print('Error: rollingJoint_fr not found')
```

Рисунок 27-Листинг кода 1 часть



```python
    return_code, wheel_joints_handle[1] = vrep_sim.simxGetObjectHandle(clientID, 'rollingJoint_fr', vrep_sim.simx_opmode_blocking)
    if return_code != vrep_sim.simx_return_ok:
        print('Error: rollingJoint_fr not found')
        sys.exit(1)
    print('get object youBot rollingJoint_fr ok.')

    return_code, wheel_joints_handle[2] = vrep_sim.simxGetObjectHandle(clientID, 'rollingJoint_rr', vrep_sim.simx_opmode_blocking)
    if return_code != vrep_sim.simx_return_ok:
        print('Error: rollingJoint_rr not found')
        sys.exit(1)
    print('get object youBot rollingJoint_rr ok.')

    return_code, wheel_joints_handle[3] = vrep_sim.simxGetObjectHandle(clientID, 'rollingJoint_rl', vrep_sim.simx_opmode_blocking)
    if return_code != vrep_sim.simx_return_ok:
        print('Error: rollingJoint_rl not found')
        sys.exit(1)
    print('get object youBot rollingJoint_rl ok.')

    # Arm joints
    arm_joints_handle = [-1,-1,-1,-1,-1]
    for i in range(5):
        return_code, arm_joints_handle[i] = vrep_sim.simxGetObjectHandle(clientID, 'Joint' + str(i+1), vrep_sim.simx_opmode_blocking)
        if return_code != vrep_sim.simx_return_ok:
            print(f'Error: Joint {i+1} not found')
            sys.exit(1)
        print(f'get object arm joint {i+1} ok.')

    # Gripper
    return_code, gripper_joint_1_handle = vrep_sim.simxGetObjectHandle(clientID, 'youBotGripperJoint1', vrep_sim.simx_opmode_blocking)
    if return_code != vrep_sim.simx_return_ok:
        print('Error: youBotGripperJoint1 not found')
        sys.exit(1)
    print('get object gripper joint 1 ok')

    return_code, gripper_joint_2_handle = vrep_sim.simxGetObjectHandle(clientID, 'youBotGripperJoint2', vrep_sim.simx_opmode_blocking)
    if return_code != vrep_sim.simx_return_ok:
        print('Error: youBotGripperJoint2 not found')
        sys.exit(1)
    print('get object gripper joint 2 ok')

    return_code, gripper_tip_handle = vrep_sim.simxGetObjectHandle(clientID, 'youBot_positionTip', vrep_sim.simx_opmode_blocking)
    if return_code != vrep_sim.simx_return_ok:
        print('Error: youBot_positionTip not found')
        sys.exit(1)
    print('get object gripper position tip ok')

    # Initialize arm joints
    desired_arm_joint_angles = [0, 0, 0, 0, 0]
```

Рисунок 28-Листинг кода 2 часть



```
    desired_arm_joint_angles = [0, 0, 0, 0, 0]
    for i in range(5):
        vrep_sim.simxSetJointPosition(clientID, arm_joints_handle[i], desired_arm_joint_angles[i], vrep_sim.simx_opmode_blocking)

    # World joints
    world_joints_handle = [-1, -1, -1]
    return_code, world_joints_handle[0] = vrep_sim.simxGetObjectHandle(clientID, 'World_X_Joint', vrep_sim.simx_opmode_blocking)
    if return_code != vrep_sim.simx_return_ok:
        print('Error: World_X_Joint not found')
        sys.exit(1)
    print('get object world joint X ok.')

    return_code, world_joints_handle[1] = vrep_sim.simxGetObjectHandle(clientID, 'World_Y_Joint', vrep_sim.simx_opmode_blocking)
    if return_code != vrep_sim.simx_return_ok:
        print('Error: World_Y_Joint not found')
        sys.exit(1)
    print('get object world joint Y ok.')

    return_code, world_joints_handle[2] = vrep_sim.simxGetObjectHandle(clientID, 'World_Th_Joint', vrep_sim.simx_opmode_blocking)
    if return_code != vrep_sim.simx_return_ok:
        print('Error: World_Th_Joint not found')
        sys.exit(1)
    print('get object world joint Th ok.')

    # Cubes
    return_code, cube_1_handle = vrep_sim.simxGetObjectHandle(clientID, 'Cube_1', vrep_sim.simx_opmode_blocking)
    if return_code != vrep_sim.simx_return_ok:
        print('Error: Cube_1 not found')
        sys.exit(1)
    print('get object cube_1 ok.')

    return_code, cube_2_handle = vrep_sim.simxGetObjectHandle(clientID, 'Cube_2', vrep_sim.simx_opmode_blocking)
    if return_code != vrep_sim.simx_return_ok:
        print('Error: Cube_2 not found')
        sys.exit(1)
    print('get object cube_2 ok.')

    return_code, cube_goal_handle = vrep_sim.simxGetObjectHandle(clientID, 'Cube_goal', vrep_sim.simx_opmode_blocking)
    if return_code != vrep_sim.simx_return_ok:
        print('Error: Cube_goal not found')
        sys.exit(1)
    print('get object cube_goal ok.')
else:
    print ('Failed connecting to remote API server')
    sys.exit(1)

# Close connection
vrep_sim.simxFinish(clientID)
```

Рисунок 29-Листинг кода 3 часть

Первоначально, не считая подключения библиотек и пользовательских модулей, я выполнил подключение к серверу Compeliasim и его инициализацию:
vrep_sim.simxFinish(-1)
clientID = vrep_sim.simxStart('127.0.0.1',19999,True,True,5000,5)

А также получение обработчиков объектов в симуляции эти обработчики нужны для того, чтобы в дальнейшем можно было взаимодействовать с объектами в симуляции
return_code, youBot_handle = vrep_sim.simxGetObjectHandle(clientID, 'youBot', vrep_sim.simx_opmode_blocking).
И в коде не мало важная роль выделенна ошибкам то есть если какая то часть робота не будет правильно инициализирована то код остановится это сделанно впервую очередь для коректной работы.Далее в коде можно выделить основные алгоритмы управления:

1.Алгоритм управления манипулятора. Этот алгоритм задаёт начальные углы для суставов манипулятора, чтобы установить его в нейтральное положение.
desired_arm_joint_angles = [0, 0, 0, 0, 0]
for i in range: vrep_sim.simxSetJointPosition(clientID,arm_joints_handle[i], desired_arm_joint_angles[i], vrep_sim.simx_opmode_blocking)

2.Алгоритм управления захватом этот алгоритм управляет захватом объектов с помощью манипулятора робота, используя суставы гриппера.

return_code, gripper_joint_1_handle = vrep_sim.simxGetObjectHandle(clientID, 'youBotGripperJoint1', vrep_sim.simx_opmode_blocking)
return_code, gripper_joint_2_handle = vrep_sim.simxGetObjectHandle(clientID, 'youBotGripperJoint2', vrep_sim.simx_opmode_blocking)

3.Алгоритм управления движением робота .Этот алгоритм управляет движением робота, устанавливая скорости вращения колёс.
vrep_sim.simxSetJointTargetVelocity(clientID, wheel_joints_handle[i], velocity, vrep_sim.simx_opmode_oneshot)
Изображения работы робота и алгоритмов:

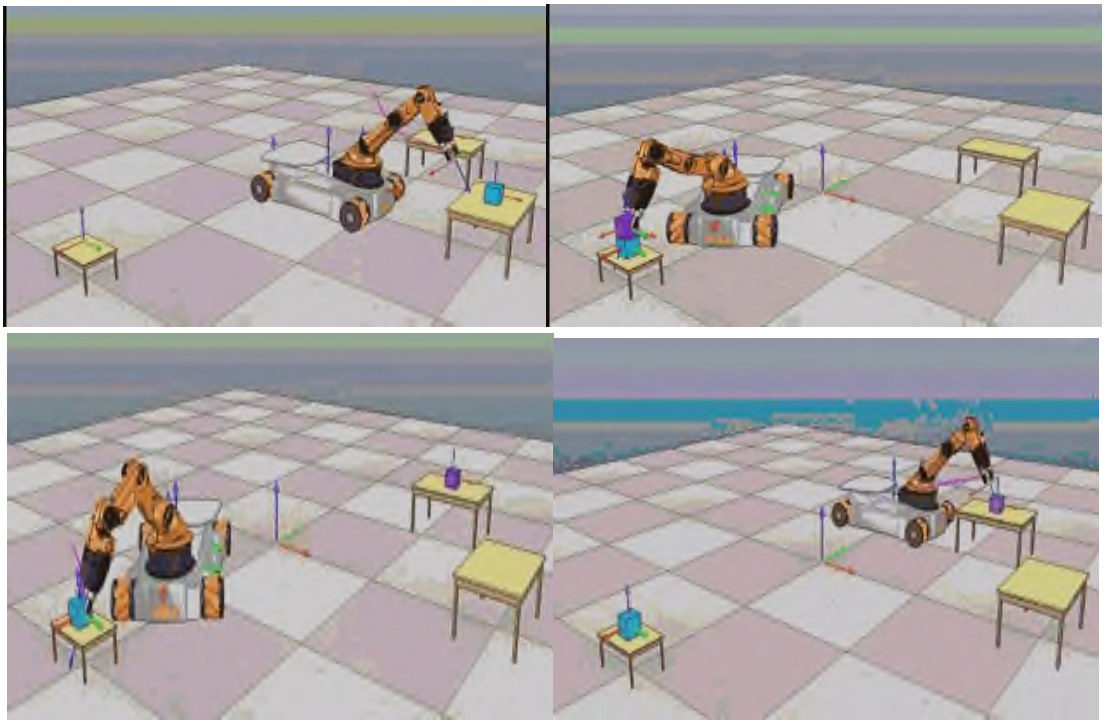


Рисунок 30 - Выполнение алгоритмов

В ходе выполнения работы я выявил основные преимущества работы не с физическим прототипом, а в виртуальной среде. Среди них стоит особо выделить максимальную точность моделирования по сравнению с реальной жизнью, а также отсутствие ответственности за повреждение дорогостоящих прототипов. Виртуальная среда обладает огромным преимуществом в полноценном тестировании и разработке, и должна использоваться перед реальными испытаниями для тщательной проверки и настройки алгоритмов, что позволяет минимизировать убытки и потери. В ходе экспериментов использовался робот Kuka Youbot, стоимость которого на официальном сайте может достигать 50 тысяч долларов, тогда как мой метод не требует финансовых затрат и может быть

реализован без единого доллара вложений. Такой подход значительно снижает риски и экономит ресурсы, что подтверждается современными исследованиями и практикой применения виртуальных симуляторов в робототехнике

ЗАКЛЮЧЕНИЕ

В этой дипломной работе я погрузился в тему использования виртуальной реальности — VR — чтобы помочь роботам работать лучше. Виртуальная реальность — это такой крутой инструмент, который позволяет создавать безопасную и удобную среду для тестирования разных алгоритмов управления роботами. Это значит, что можно проверить, как робот себя поведёт, не ломая реальное железо и не тратя кучу денег. Главная задача была — сделать виртуальную среду, в которой можно было бы пробовать разные алгоритмы, чтобы понять, насколько они надёжные и эффективные. В теории я подробно рассмотрел, как VR и искусственный интеллект могут работать вместе, и как виртуальная реальность помогает сделать управление роботами лучше. Практическая часть заключалась в том, что я создал виртуальных роботов и реализовал алгоритм «Возьми и поставь» — то есть проверял, как робот в VR умеет брать и перемещать объекты. Это помогло понять, насколько удобно и эффективно использовать VR для таких задач. По результатам работы стало ясно — виртуальная реальность действительно облегчает жизнь. Она сокращает риски, экономит время и деньги, которые обычно уходят на тесты с настоящими роботами. VR открывает большие возможности для того, чтобы улучшать алгоритмы и делать роботов умнее. В итоге, я убедился, что VR — это не просто модный тренд, а очень перспективный инструмент для робототехники. Он помогает разработчикам создавать и проверять алгоритмы быстрее, проще и безопаснее — а это сейчас очень важно.

СПИСОК ТЕРМИНОВ И СОКРАЩЕНИЙ

1. **VR (Virtual Reality)** — Виртуальная реальность. Технология, создающая искусственную среду с использованием компьютерных технологий, в которой пользователь может взаимодействовать с объектами.
2. **Искусственный интеллект (ИИ)** — Совокупность методов и технологий, направленных на создание систем, которые могут выполнять задачи, требующие человеческого интеллекта, такие как распознавание образов, обработка естественного языка и принятие решений.
3. **Робототехника** — Область науки и техники, занимающаяся разработкой роботов, их проектированием и применением в различных сферах.
4. **Алгоритмы управления** — Совокупность инструкций, предназначенных для управления движением или поведением робота, включая манипуляции с объектами и навигацию.
5. **Симуляция** — Процесс создания и использования виртуальной модели реальной системы для анализа её поведения в различных условиях.
6. **Тестирование алгоритмов управления** — Процесс проверки и оценки работы алгоритмов управления в различных сценариях с целью выявления ошибок и улучшения их эффективности.
7. **Моделирование** — Создание виртуальной модели объекта или процесса, которое позволяет исследовать его поведение в различных условиях без использования реального оборудования.
8. **Обратная связь** — Процесс получения информации о текущем состоянии системы с целью корректировки её работы в реальном времени.
9. **Алгоритм «Возьми и поставь»** — Задача манипуляции, при которой робот должен захватить объект и переместить его в заранее заданную точку.
10. **Робот-манипулятор** — Робот с механическими руками, использующимися для захвата, перемещения и манипуляции объектами.
11. **VREP (Virtual Robot Experimentation Platform)** — Платформа для создания и тестирования виртуальных моделей роботов и алгоритмов управления, поддерживающая взаимодействие с различными программными инструментами и языками программирования.
12. **KPI (Key Performance Indicator)** — Ключевые показатели эффективности. Параметры, которые используются для оценки успешности алгоритмов или действий робота.

13. **Интерфейс программирования приложений (API)** — Набор инструментов и протоколов, который позволяет программным компонентам взаимодействовать друг с другом.
14. **Обработка данных с датчиков** — Процесс сбора и анализа информации, поступающей с различных датчиков робота для корректировки его поведения.
15. **Технологии оптимизации** — Методы и подходы, используемые для улучшения алгоритмов с целью повышения их эффективности и производительности в реальных условиях.

СПИСОК ИСПОЛЬЗОВАННОЙ ЛИТЕРАТУРЫ

[1] Виртуальная реальность // Википедия: свободная энциклопедия. – Режим доступа: https://ru.wikipedia.org/wiki/Виртуальная_реальность [2] Ломая барьеры: как VR-робототехника создает будущее профессионального сотрудничества // Securities.io. – Режим доступа: https://www.securities.io/ru/breaking-barriers-how-vr-robotics-is-crafting-the-future-of-seamless-professional-collaboration/
[3] Робототехника и VR: интеграция технологий будущего // VR-App.ru. – Режим доступа: https://vr-app.ru/blog/robototexnika-i-vr-integraciia-texnologii/

[4] Роботы, искусственный интеллект, дополненная и виртуальная реальность: этические, правовые и гигиенические проблемы // CyberLeninka. – Режим доступа: https://cyberleninka.ru/article/n/roboty-iskusstvennyy-intellekt-dopolnennaya-i-virtualnaya-realnost-eticheskie-pravovye-i-gigienicheskie-problemy
[5] Применение VR-технологий в робототехнике: разработка виртуальных моделей и управление ими // VR-App.ru. – Режим доступа: https://vr-app.ru/blog/primenenie-vr-texnologii-v-robototexnike-razrabotka-virtualnyx-modelei-i-upravlenie-imi/
[6] Искусственный интеллект, виртуальная реальность и робототехника // MedicalExpo.ru. – Режим доступа: https://www.medicalexpo.ru/cat/iskusstvennyj-intellekt-virtual-naa-real-nost-robototehnika-HN.html
[7]Алгоритмы управления роботами https://xn--18-6kcdusowgbt1a4b.xn--p1ai/%D0%B0%D0%BB%D0%B3%D0%BE%D1%80%D0%B8%D1%82%D0%BC%D1%8B-%D0%B2-%D1%80%D0%BE%D0%B1%D0%BE%D1%82%D0%BE%D1%82%D0%B5%D1%85%D0%BD%D0%B8%D0%BA%D0%B5
[8] Алгоритмы и системы дистанционного управления роботами: теория и практика // ЛитРес. https://www.litres.ru/book/inzhener-33334081/algoritmy-i-sistemy-distancionnogo-upravleniya-robotami-71951098/
[9] Сенотов В.Д., Алисейчик А.П., Павловский Е.В., Подопросветов А.В., Орлов И.А. Алгоритмы стайного децентрализованного управления движением группы роботов с дифференциальным приводом // Препринты ИПМ им. М.В. Келдыша. – 2020. – № 123. – 39 с. – Режим доступа: https://library.keldysh.ru/preprint.asp?id=2020-123
[10] Lynch K.M., Park F.C. Modern Robotics: Mechanics, Planning, and Control. – Cambridge: Cambridge University Press, 2017. https://hades.mech.northwestern.edu/images/7/7f/MR.pdf

[11] Староверов А. Улучшаем навигацию роботов с помощью нейронного потенциального поля // Habr.com. – Режим доступа: https://habr.com/ru/companies/airi/articles/844682/
[12] Динамика мирового рынка AR-VR: https://www.huawei.ru/news/issledovanie-huawei-rynok-ar-vr-v-rossii-dostignet-7-mlrd-rub-k-2025-godu/
[13] Компоненты VR https://www.biztechcs.com/how-to-make-vr-app/
[14] Интрефейс программы https://mde.tw/pjcopsim/content/userInterface.html

ПРИЛОЖЕНИЕ А:

```
import numpy as np import modern_robotics as mr

def trajectory_generator(T_se_initial, T_sc_initial, T_sc_final, T_ce_grasp,
T_ce_standoff, k): # Сегмент 1: Перемещение в standby-позицию над объектом
T_se_standoff_initial = T_sc_initial @ T_ce_standoff T_se_segment1 =
mr.CartesianTrajectory(T_se_initial, T_se_standoff_initial, 5, 500, 3)

# Сегмент 2: Спуск к объекту
T_se_grasp = T_sc_initial @ T_ce_grasp
T_se_segment2 = mr.CartesianTrajectory(T_se_segment1[-1], T_se_grasp, 2, 200, 3)
T_se_before = np.append(T_se_segment1, T_se_segment2, axis=0)

# Сегмент 3: Закрытие захвата (пауза 0.64 сек)
T_se_before = np.append(T_se_before, np.tile(T_se_before[-1], (64, 1)), axis=0)

# Сегмент 4: Возврат в standby-позицию
T_se_segment4 = mr.CartesianTrajectory(T_se_grasp, T_se_standoff_initial, 2, 200,
3)
T_se_before = np.append(T_se_before, T_se_segment4, axis=0)

# Сегмент 5: Перемещение к целевой standby-позиции
T_se_standoff_final = T_sc_final @ T_ce_standoff
T_se_segment5 = mr.CartesianTrajectory(T_se_standoff_initial, T_se_standoff_final,
8, 800, 3)
T_se_before = np.append(T_se_before, T_se_segment5, axis=0)

# Сегмент 6: Спуск в конечную позицию
T_se_lose = T_sc_final @ T_ce_grasp
T_se_segment6 = mr.CartesianTrajectory(T_se_standoff_final, T_se_lose, 2, 200, 3)
T_se_before = np.append(T_se_before, T_se_segment6, axis=0)

# Сегмент 7: Открытие захвата (пауза 0.64 сек)
T_se_before = np.append(T_se_before, np.tile(T_se_before[-1], (64, 1)), axis=0)

# Сегмент 8: Возврат в standby-позицию
T_se_segment8 = mr.CartesianTrajectory(T_se_before[-1], T_se_standoff_final, 2,
200, 3)
```

```
T_se_before = np.append(T_se_before, T_se_segment8, axis=0)

# Формирование выходной матрицы
total_steps = int(k * 21 / 0.01 + 64 * 2)
T_se_post = np.zeros((total_steps, 13))

for i in range(total_steps):
    T_se_post[i, :3] = T_se_before[i, :3, 3]  # Позиция (x, y, z)
    T_se_post[i, 3:6] = T_se_before[i, 0, :3]  # Ориентация (r, p, y)
    T_se_post[i, 6:9] = T_se_before[i, 1, :3]  # Ориентация (r, p, y)
    T_se_post[i, 9:12] = T_se_before[i, 2, :3]  # Ориентация (r, p, y)
    T_se_post[i, 12] = 0

# Активация захвата на этапах удержания объекта
grasp_start = int(k * 7 / 0.01)
grasp_end = int(k * 19 / 0.01 + 64)
T_se_post[grasp_start:grasp_end, 12] = 1

return T_se_post
```

ПРИЛОЖЕНИЕ В:

```
import numpy as np import sys
sys.path.append(r"C:\Users_adam\OneDrive\Desktop\код")

from TrajectoryGenerator import trajectoryGenerator

from NextState import nextState

from FeedbackControl import feedbackControl

import time import sys sys.path.append('./vrep/VREP_remoteAPIs')

try: import remoteApi as vrep_sim except: print ('--------------------------------------------
------------------')

print ('"remoteApi.py" could not be imported. Make sure it is in the same folder as this
file.')

print ('------------------------------------------------------------')

print ('Program started') vrep_sim.simxFinish(-1) clientID =
vrep_sim.simxStart('127.0.0.1',19999,True,True,5000,5) if clientID != -1: print
('Connected to remote API server')
return_code, youBot_handle = vrep_sim.simxGetObjectHandle(clientID, 'youBot',
vrep_sim.simx_opmode_blocking)                                    if return_code
!=      vrep_sim.simx_return_ok:
   print('Error: Object youBot not found')
   sys.exit(1)
print('get object youBot ok.')
return_code, youBot_dummy_handle = vrep_sim.simxGetObjectHandle(clientID,
'youBotDummy', vrep_sim.simx_opmode_blocking)
if return_code != vrep_sim.simx_return_ok:
   print('Error: Object youBotDummy not found')
   sys.exit(1)
print('get object youBotDummy ok.')
wheel_joints_handle = [-1,-1,-1,-1]
return_code, wheel_joints_handle[0] = vrep_sim.simxGetObjectHandle(clientID,
'rollingJoint_fl', vrep_sim.simx_opmode_blocking)
if return_code != vrep_sim.simx_return_ok:
```

```
    print('Error: rollingJoint_fl not found')
    sys.exit(1)
print('get object youBot rollingJoint_fl ok.')

return_code, wheel_joints_handle[1] = vrep_sim.simxGetObjectHandle(clientID,
'rollingJoint_fr', vrep_sim.simx_opmode_blocking)
if return_code != vrep_sim.simx_return_ok:
    print('Error: rollingJoint_fr not found')
    sys.exit(1)
print('get object youBot rollingJoint_fr ok.')

return_code, wheel_joints_handle[2] = vrep_sim.simxGetObjectHandle(clientID,
'rollingJoint_rr', vrep_sim.simx_opmode_blocking)
if return_code != vrep_sim.simx_return_ok:
    print('Error: rollingJoint_rr not found')
    sys.exit(1)
print('get object youBot rollingJoint_rr ok.')

return_code, wheel_joints_handle[3] = vrep_sim.simxGetObjectHandle(clientID,
'rollingJoint_rl', vrep_sim.simx_opmode_blocking)
if return_code != vrep_sim.simx_return_ok:
    print('Error: rollingJoint_rl not found')
    sys.exit(1)
print('get object youBot rollingJoint_rl ok.')

arm_joints_handle = [-1,-1,-1,-1,-1]
for i in range(5):
    return_code, arm_joints_handle[i] = vrep_sim.simxGetObjectHandle(clientID,
'Joint' + str(i+1), vrep_sim.simx_opmode_blocking)
    if return_code != vrep_sim.simx_return_ok:
        print(f'Error: Joint {i+1} not found')
        sys.exit(1)
    print(f'get object arm joint {i+1} ok.')


return_code, gripper_joint_1_handle = vrep_sim.simxGetObjectHandle(clientID,
'youBotGripperJoint1', vrep_sim.simx_opmode_blocking)
```

```
if return_code != vrep_sim.simx_return_ok:
    print('Error: youBotGripperJoint1 not found')
    sys.exit(1)
print('get object gripper joint 1 ok')

return_code, gripper_joint_2_handle = vrep_sim.simxGetObjectHandle(clientID,
'youBotGripperJoint2', vrep_sim.simx_opmode_blocking)
if return_code != vrep_sim.simx_return_ok:
    print('Error: youBotGripperJoint2 not found')
    sys.exit(1)
print('get object gripper joint 2 ok')

return_code, gripper_tip_handle = vrep_sim.simxGetObjectHandle(clientID,
'youBot_positionTip', vrep_sim.simx_opmode_blocking)
if return_code != vrep_sim.simx_return_ok:
    print('Error: youBot_positionTip not found')
    sys.exit(1)
print('get object gripper position tip ok')
```

ПРИЛОЖЕНИЕ С:

```
desired_arm_joint_angles = [0, 0, 0, 0, 0]
for i in range(5):
  vrep_sim.simxSetJointPosition(clientID, arm_joints_handle[i],
desired_arm_joint_angles[i], vrep_sim.simx_opmode_blocking)
world_joints_handle = [-1, -1, -1]
return_code, world_joints_handle[0] = vrep_sim.simxGetObjectHandle(clientID,
'World_X_Joint', vrep_sim.simx_opmode_blocking)
if return_code != vrep_sim.simx_return_ok:
   print('Error: World_X_Joint not found')
   sys.exit(1)
print('get object world joint X ok.')
return_code, world_joints_handle[1] = vrep_sim.simxGetObjectHandle(clientID,
'World_Y_Joint', vrep_sim.simx_opmode_blocking)
if return_code != vrep_sim.simx_return_ok:
   print('Error: World_Y_Joint not found')
   sys.exit(1)
print('get object world joint Y ok.')
return_code, world_joints_handle[2] = vrep_sim.simxGetObjectHandle(clientID,
'World_Th_Joint', vrep_sim.simx_opmode_blocking)
if return_code != vrep_sim.simx_return_ok:
   print('Error: World_Th_Joint not found')
   sys.exit(1)
print('get object world joint Th ok.')
return_code, cube_1_handle = vrep_sim.simxGetObjectHandle(clientID, 'Cube_1',
vrep_sim.simx_opmode_blocking)
if return_code != vrep_sim.simx_return_ok:
   print('Error: Cube_1 not found')
   sys.exit(1)
print('get object cube_1 ok.')
return_code, cube_2_handle = vrep_sim.simxGetObjectHandle(clientID, 'Cube_2',
vrep_sim.simx_opmode_blocking)
if return_code != vrep_sim.simx_return_ok:
   print('Error: Cube_2 not found')
   sys.exit(1)
print('get object cube_2 ok.')
```

```
return_code, cube_goal_handle = vrep_sim.simxGetObjectHandle(clientID,
'Cube_goal', vrep_sim.simx_opmode_blocking)
if return_code != vrep_sim.simx_return_ok:
    print('Error: Cube_goal not found')
    sys.exit(1)
print('get object cube_goal ok.')
 else: print ('Failed connecting to remote API server')
sys.exit(1)vrep_sim.simxFinish(clientID)
```

НЕКОММЕРЧЕСКОЕ АКЦИОНЕРНОЕ ОБЩЕСТВО «КАЗАХСКИЙ НАЦИОНАЛЬНЫЙ ИССЛЕДОВАТЕЛЬСКИЙ ТЕХНИЧЕСКИЙ УНИВЕРСИТЕТ имени К.И.САТПАЕВА»

РЕЦЕНЗИЯ

на дипломный проект (работе)
Бейсултанов Адам Русланович

6В07113 – «Робототехника и мехатроника»

На тему: Анализ применения VR для тестирования и оптимизации алгоритмов управления роботами в различных сценариях без необходимости физического прототипа.

Выполнено:
а) графическая часть на 23 листах
б) пояснительная записка на 58 страницах

ЗАМЕЧАНИЯ К РАБОТЕ

Дипломная работа посвящена актуальной и перспективной теме — применению технологий виртуальной реальности (VR) в процессе разработки и отладки алгоритмов управления роботами. В условиях стремительного развития робототехники, где требуется высокая степень адаптивности и гибкости, использование VR становится мощным инструментом для моделирования и анализа поведения роботов в различных ситуациях без необходимости физического прототипирования.

Работа логично структурирована и включает как теоретическую, так и практическую части. В теоретической части автор провёл всесторонний анализ существующих VR-решений, применяемых в робототехнике, и обоснованно сравнил их с традиционными методами. Особо стоит отметить акцент на таких преимуществах виртуальной среды, как гибкость настройки сценариев, безопасность экспериментов, возможность многократного повторения и применения методов обучения с подкреплением.

Практическая часть представлена серией моделируемых экспериментов в виртуальной среде CoppeliaSim, что демонстрирует не только владение современными инструментами, но и умение применять теоретические знания на практике. Студент подробно описывает, как изменения параметров влияют на эффективность алгоритмов управления, и делает обоснованные выводы по результатам наблюдений.

Результатом работы является демонстрация того, что внедрение VR в процесс разработки и тестирования управляемых систем роботов позволяет существенно сократить временные и ресурсные затраты, повысить безопасность и масштабируемость исследований, а также способствует генерации инновационных решений.

Оценка работы

Работа отличается актуальностью, высоким уровнем проработки материала, грамотной подачей и обоснованными выводами. Рекомендую работу к защите с оценкой 90.

Рецензент
Ассоциированный профессор
Академия гражданской авиации
_____ Сейдилдаева А.К.
(подпись)
«30» мая 2025 г.

Ф КазНИТУ 706-17. Рецензия

НЕКОММЕРЧЕСКОЕ АКЦИОНЕРНОЕ ОБЩЕСТВО «КАЗАХСКИЙ НАЦИОНАЛЬНЫЙ ИССЛЕДОВАТЕЛЬСКИЙ ТЕХНИЧЕСКИЙ УНИВЕРСИТЕТ имени К.И.САТПАЕВА»

ОТЗЫВ

на дипломный проект (работе)
Бейсултанов Адам Русланович
6В07113 - " Робототехника и мехатроника»

на тему: Анализ применения VR для тестирования и оптимизации алгоритмов управления роботами в различных сценариях без необходимости физического прототипа.

Дипломная работа посвящена перспективному направлению в области робототехники — использованию технологий виртуальной реальности (VR) для тестирования и оптимизации алгоритмов управления роботами. В условиях стремительного развития интеллектуальных систем и необходимости быстрого и безопасного прототипирования, исследование данной темы является особенно актуальным.

Студент Бейсултанов Адам проявил искренний интерес к исследуемой проблематике, что отразилось как в глубоком теоретическом анализе, так и в практической реализации. В теоретической части грамотно раскрыта сущность VR-технологий и их синергия с методами искусственного интеллекта. Особое внимание уделено преимуществам виртуальной среды при проектировании и проверке алгоритмов — таким как снижение затрат, повышение гибкости моделирования и обеспечение безопасности экспериментов.

Практическая часть работы заслуживает высокой оценки. Студент Бейсултанов А. разработал и реализовал в среде CoppeliaSim модель виртуального робота с возможностью выполнения манипуляций типа «возьми и поставь». Это позволило на практике протестировать различные сценарии и убедиться в эффективности применяемых алгоритмов управления. Работа выполнена с хорошим техническим уровнем, а сделанные выводы подтверждены экспериментальными результатами.

Следует отметить, что Бейсултанов А. продемонстрировал самостоятельность, высокий уровень мотивации, ответственность и творческий подход к поставленной задаче. Работа носит практическую направленность и может быть полезна для дальнейших исследований и разработок в области робототехники.

Считаю, что дипломная работа соответствует требованиям, и заслуживает оценки «90» и заслуживает присуждению степени бакалавра.

Дипломный руководитель
Старший преподаватель каф.РТиТСА

_____ Байтуганова В.К.
(подпись)

«5» 06 2025 г.

Отчет подобия

Метаданные

Название организации
Satbayev University
Название
Тестирование алгоритмов управления в VR среде без необходимости физического прототипа

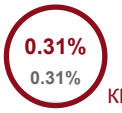
Автор
Бейсултанов Адам Русланович
Научный руководитель / Эксперт
Венера Байтурганова

Подразделение
ИАиИТ

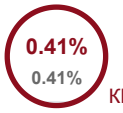
Объем найденных подобий

КП-ия определяют, какой процент текста по отношению к общему объему текста был найден в различных источниках.. Обратите внимание!Высокие значения коэффициентов не означают плагиат. Отчет должен быть проанализирован экспертом.

КП1



КП2



КЦ

25
Длина фразы для коэффициента подобия 2

8034
Количество слов

63310
Количество символов

Тревога

В этом разделе вы найдете информацию, касающуюся текстовых искажений. Эти искажения в тексте могут говорить о ВОЗМОЖНЫХ манипуляциях в тексте. Искажения в тексте могут носить преднамеренный характер, но чаще, характер технических ошибок при конвертации документа и его сохранении, поэтому мы рекомендуем вам подходить к анализу этого модуля со всей долей ответственности. В случае возникновения вопросов, просим обращаться в нашу службу поддержки.

Замена букв		0
Интервалы		0
Микропробелы		6
Белые знаки		0
Парафразы (SmartMarks)	a	8

Подобия по списку источников

Ниже представлен список источников. В этом списке представлены источники из различных баз данных. Цвет текста означает в каком источнике он был найден. Эти источники и значения Коэффициента Подобия не отражают прямого плагиата. Необходимо открыть каждый источник и проанализировать сожержание и правильность оформления источника.

10 самых длинных фраз — Цвет текста

ПОРЯДКОВЫЙ НОМЕР	НАЗВАНИЕ И АДРЕС ИСТОЧНИКА URL (НАЗВАНИЕ БАЗЫ)	КОЛИЧЕСТВО ИДЕНТИЧНЫХ СЛОВ (ФРАГМЕНТОВ)
1	https://www.freesion.com/article/24981405510/	25 0.31 %
2	https://thinkingneuron.com/how-to-find-best-hyperparameters-using-gridsearchcv-in-python/	19 0.24 %
3	https://scilead.ru/article/5407-intellektualnie-sistemi-upravleniya-vozdushni	15 0.19 %
4	https://infourok.ru/konspekt-uroka-tehnologii-na-temu-nashi-proekti-akvarium-klass-3962592.html	13 0.16 %

5	https://cyberleninka.ru/article/n/razrabotka-novyh-interfeysov-hci-dlya-bolee-estestvennogo-i-intuitivnogo-vzaimodeystviya-s-kompyuterami	12 0.15 %
6	Projekt zasilania domu jednorodzinnego zlokalizowanego w Rudce za pomocą ogniw fotowoltaicznych [redacted] 4/26/2018 Politechnika Lubelska (Politechnika Lubelska)	12 0.15 %
7	https://www.freesion.com/article/24981405510/	12 0.15 %
8	http://www.dut.edu.ua/uploads/l_1299_74309159.doc	11 0.14 %
9	http://www.dut.edu.ua/uploads/l_1299_74309159.doc	10 0.12 %
10	https://infourok.ru/konspekt-uroka-tehnologii-na-temu-nashi-proekti-akvarium-klass-3962592.html	9 0.11 %

из базы данных RefBooks (0.00 %)

ПОРЯДКОВЫЙ НОМЕР	НАЗВАНИЕ	КОЛИЧЕСТВО ИДЕНТИЧНЫХ СЛОВ (ФРАГМЕНТОВ)

из домашней базы данных (0.00 %)

ПОРЯДКОВЫЙ НОМЕР	НАЗВАНИЕ	КОЛИЧЕСТВО ИДЕНТИЧНЫХ СЛОВ (ФРАГМЕНТОВ)

из программы обмена базами данных (0.15 %)

ПОРЯДКОВЫЙ НОМЕР	НАЗВАНИЕ	КОЛИЧЕСТВО ИДЕНТИЧНЫХ СЛОВ (ФРАГМЕНТОВ)
1	Projekt zasilania domu jednorodzinnego zlokalizowanego w Rudce za pomocą ogniw fotowoltaicznych [redacted] 4/26/2018 Politechnika Lubelska (Politechnika Lubelska)	12 (1) 0.15 %

из интернета (1.85 %)

ПОРЯДКОВЫЙ НОМЕР	ИСТОЧНИК URL	КОЛИЧЕСТВО ИДЕНТИЧНЫХ СЛОВ (ФРАГМЕНТОВ)
1	**https://www.freesion.com/article/24981405510/**	**37 (2) 0.46 %**
2	https://infourok.ru/konspekt-uroka-tehnologii-na-temu-nashi-proekti-akvarium-klass-3962592.html	35 (4) 0.44 %
3	http://www.dut.edu.ua/uploads/l_1299_74309159.doc	31 (4) 0.39 %
4	https://thinkingneuron.com/how-to-find-best-hyperparameters-using-gridsearchcv-in-python/	19 (1) 0.24 %
5	https://scilead.ru/article/5407-intellektualnie-sistemi-upravleniya-vozdushni	15 (1) 0.19 %
6	https://cyberleninka.ru/article/n/razrabotka-novyh-interfeysov-hci-dlya-bolee-estestvennogo-i-intuitivnogo-vzaimodeystviya-s-kompyuterami	12 (1) 0.15 %

Список принятых фрагментов (нет принятых фрагментов)

ПОРЯДКОВЫЙ НОМЕР	СОДЕРЖАНИЕ	КОЛИЧЕСТВО ИДЕНТИЧНЫХ СЛОВ (ФРАГМЕНТОВ)