Министерство науки и высшего образования Российской Федерации
Московский физико-технический институт
(национальный исследовательский университет)
Физтех-школа прикладной математики и информатики
Магистерская программа
«Методы и технологии искусственного интеллекта»

Студент Рублева Елена Алексеевна

Разработка методов коррекции траектории движения робототехнического манипулятора

09.04.01 — Информатика и вычислительная техника

Магистерская диссертация

Научный руководитель:
старший научный сотрудник, PhD
Миронов К. В.

Москва
2025 г.

Содержание

1 Введение **4**

2 Обзорно-постановочный раздел работы **5**

2.1 Основные понятия и определения 5

2.2 Обзор современного состояния проблемы 12

2.3 Формальная постановка задачи 26

3 Основной раздел работы **28**

3.1 Базовое решение 29

3.2 Предлагаемый метод 30

3.3 Представление препятствий 34

3.4 Генерация начальной траектории 36

4 Вычислительные эксперименты **57**

4.1 Эксперименты, связанные с представлением препятствий 58

4.2 Эксперименты, связанные с генерацией начальной траектории 60

5 Заключение **72**

Аннотация

В работе исследуется задача планирования траекторий для роботизированных манипуляторов в условиях окружающей среды, в которой расположены препятствия. Цель работы заключается в разработке нового подхода, который сочетает алгоритм TrajOpt с методами глубокого обучения, позволяя улучшить эффективность и точность планирования движений. В результате исследования разработаны и реализованы алгоритмы, способные корректировать траектории на основе входных данных с сенсоров, что обеспечивается высоким уровнем адаптивности системы. Проведенные вычислительные эксперименты показали, что предложенные методы позволяют существенно сократить время на расчет траекторий и повысить их надежность в условиях присутствия препятствий. Также обсуждаются перспективы применения разработанных методов для коллаборативных роботов, что подчеркивает их значимость для дальнейшего совершенствования робототехнических систем. Работа дает новое понимание путей повышения эффективности манипуляторов и открывает направления для будущих исследований в данной области.

1 Введение

В условиях стремительного развития технологий и увеличения применения автоматизированных систем, задача эффективного планирования траекторий для роботизированных манипуляторов становится все более актуальной. Современные роботы, выполняющие функции, аналогичные человеческим рукам, находят применение в различных сферах, таких как промышленность, медицина и сервисные услуги. Решение задачи построения безопасных и эффективных траекторий является критически важным, поскольку позволяет улучшить производительность роботизированных систем, а также минимизировать риски, связанные со взаимодействием роботов с окружающей средой.

До настоящего времени данная проблема решалась различными методами, включая традиционные алгоритмы, основанные на геометрическом подходе к планированию. В большинстве случаев такие алгоритмы сталкиваются с трудностями при наличии препятствий в рабочей области, что приводит к затруднениям в нахождении оптимальных траекторий. Таким образом, недостатки этих решений заключаются в их низкой эффективности и высокой временной сложности, особенно в динамически изменяющихся условиях.

В данной работе предлагается новый подход к планированию траекторий для манипуляторов с использованием алгоритма TrajOpt. Основной акцент сделан на изменение представления препятствий и интеграцию нейронных сетей, что позволяет повысить эффективность, надежность и безопасность планирования траекторий. Предлагаемые методы направлены на улучшение взаимодействия робота со средой с препятствиями и преодоление недостатков существующих решений, таких как частые ошибки в определении свободного пространства и недостаточная адаптивность к сложным условиям.

Ключевыми понятиями, необходимыми для понимания постановки задачи, являются: «робот-манипулятор» – система, выполняющая задачи, аналогичные действиям человеческой руки; «траектория» – путь, который должен пройти манипулятор для достижения заданной цели; «конфигурационное пространство с препятствиями» – область рабочего пространства, в которой находятся объекты, мешающие выполнению задачи. Цели исследования заключаются в разработке и предложении новых методов, которые повысят успешность планирования движений манипулятора в условиях наличия препятствий в окружающей среде.

В ходе исследования была проведена работа по анализу существующих подходов, по результатам которой обнаружено, что большинство алгоритмов не учитывают препятствия или работают в условиях, где роботу не нужно пересекать их. Это свидетельствует о необходимости более глубокого анализа и предложению нового подхода к планированию траекторий. Задачи исследования определяется как создание алгоритмов изменения представления препятствий и генерации начального приближения траектории.

В первом разделе обсуждаются основные понятия и определения, связанные с роботами-манипуляторами и их кинематикой. Второй раздел рассматривает нынешнее состояние проблемы и существующие методы планирования траекторий. В основном разделе работы подробно описаны предлагаемые методы, основанные на доработках алгоритма TrajOpt, а также результаты вычислительных экспериментов, проведенных для тестирования эффективности предложенных обновлений.

Основные результаты работы сводятся к улучшению алгоритма планирования, повышению скоростей расчетов и успешности построения свободных от препятствий траекторий, что особенно актуально при использовании сенсорных данных. Отмечается, что предложенные подходы, комбинация нейронных сетей и алгоритма TrajOpt, могут значительно повысить адаптивность робототехнических систем при движении в среде с препятствиями, что делает работу актуальной и своевременной.

2 Обзорно-постановочный раздел работы

В данном разделе рассмотрены основные понятия и определения, связанные с роботами-манипуляторами, а также проложен путь к пониманию их кинематики и методов планирования движений, выделены ключевые компоненты этих систем, а также опишем алгоритмы, которые применяются для их функционирования. Понимание этих базовых понятий является необходимым условием для дальнейших исследований в области кинематики и планирования траекторий, что будет предметом более глубокого анализа в следующих разделах.

2.1 Основные понятия и определения

Робот-манипулятор, также именуемый манипуляционной системой, представляет собой устройство, которое выполняет действия, аналогичные тем, что осуществ-

ляет человек с помощью рук. Однако эта аналогия не всегда точна. Важно отметить, что манипулятор выполняет функции по захвату, перемещению и установке объектов, а также может работать с различными инструментами и приспособлениями. Более того, он также способен интегрироваться с информационными системами, что позволяет ему эффективно выполнять анализ окружающей среды, принимать решения и осуществлять интенсивное планирование своих действий.

Основные компоненты робота-манипулятора включают в себя манипулятор и контроллер, который управляет действиями устройства. Манипулятор, как правило, состоит из нескольких элементов, или звеньев, которые могут вращаться вокруг своих осей. На конце манипулятора располагается рабочий орган, который выполняет широкий спектр задач. Этот рабочий орган, который также может называться захватным устройством (например, End Effector или EE), предназначен для захвата и удерживания объектов, таких как инструменты и полезные грузы.

Стоит отметить, что манипулятор включает в себя три главные подсистемы:

- исполнительный механизм, отвечающий за преобразование движения;
- рабочий орган, или захватное устройство (ЗУМ);
- система приводов, управляющая движениями манипулятора.

Исполнительный механизм обеспечивает преобразование движений отдельного звена в движения рабочего органа. Конструкция робота-манипулятора включает в себя несколько звеньев, которые соединены между собой, формируя функциональную единицу.

Соседние звенья образуют кинематические пары, представляющие собой подвижные соединения двух взаимодействующих звеньев. Каждая кинематическая пара ограничивает относительное движение между соединяемыми звеньями. Эти пары классифицируются в зависимости от количества степеней свободы, которые теряются при соединении. В этом контексте они подразделяются на пять классов, и движение соединенных элементов описывается шестью обобщенными координатами.

Для глубокого анализа кинематики робота-манипулятора необходимо определить несколько ключевых понятий:

- степень свободы — это показатель, характеризующий возможность независимого перемещения;

- число степеней свободы — это сумма независимых перемещений внутри механической системы;
- степень подвижности механизма — это общее количество обобщенных координат, которые описывают систему.

Таким образом, механизм определяется как кинематическая цепь, состоящая из неподвижного звена (часто стойки), где число степеней свободы совпадает с числом обобщенных координат, описывающих его положение относительно неподвижного звена. Движение звеньев механизма рассматривается относительно этого неподвижного элемента.

Если механизм движется в одной плоскости, он называется плоским, тогда как если его подвижные точки описывают сложные и склонные к пересечению траектории, он классифицируется как пространственный. Исполнительные механизмы робота-манипулятора, как правило, являются пространственными.

Кинематический анализ включает в себя решение двух основных задач: прямой и обратной задач кинематики.

Прямая задача кинематики (ПЗК) связана с вычислением положения и ориентации системы координат, которая прикреплена к захвату или рабочему инструменту, при известных обобщенных координатах манипулятора.

С другой стороны, обратная задача кинематики (ОЗК) заключается в определении необходимых обобщенных координат манипулятора с учетом заданных координат положения и ориентации конечной системы координат, которая связана с рабочим инструментом или захватом.

Для проведения исследования используется робот-манипулятор Universal Robots UR5. Кинематическая структура робота представлена на рисунке 1.

Для взаимодействия робота с окружающей средой используют средства очувствления. Это – системы технического зрения, датчики сил и моментов взаимодействия с внешней средой, тактильные датчики, ультразвуковые датчики, дальномеры и сканеры. Средства очувствления служат для контроля работы роботов. В современных роботах сигналы средств очувствления используются также и при управлении манипулятором.

Для исследование среды, в которой работает робот, чаще всего применяются камеры и Lidar. Нейронные сети могут обучаться на изменяющихся данных, что позволяет роботам адаптироваться к новым условиям или неожиданным ситуаци-

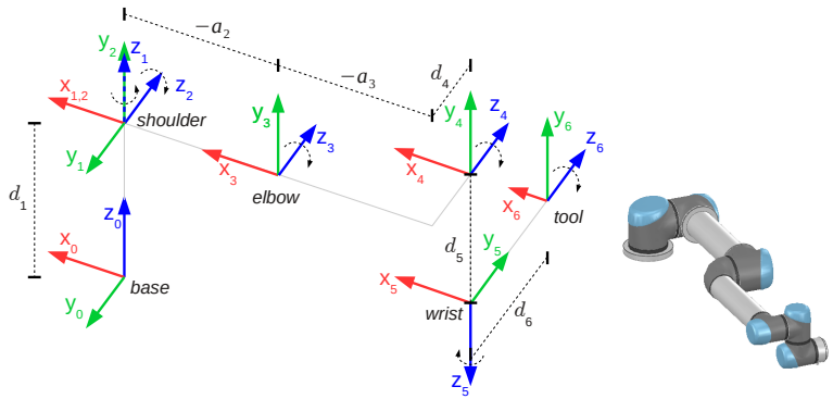


Рис. 1: Робот в стартовой позиции: слева – описание звеньев, справа – визуализация

ям в реальном времени. Это делает их особенно полезными в неструктурированных средах. Камеры фиксируют свет и цвет в окружающей среде, создавая двумерные изображения. На основе этих изображений можно извлекать информацию о формах, цветах, текстурах и движениях объектов. Для вычисления расстояния до объектов используются стереопары — устройства, состоящие из двух или более камер, позволяющие получать трехмерную информацию о окружающей среде за счет анализа разницы в изображениях. Однако у камер есть существенный недостаток — высокая чувствительность к освещенности, что делает их неэффективными в ночное время. Этого недостатка лишен Lidar, который представляет собой устройство, использующее лазеры для измерения расстояний до объектов. Он излучает лазерные импульсы и измеряет время, прошедшее до того, как импульс вернулся после отражения от объекта. На основе этих данных создается трехмерная карта окружающей среды. Однако текущий недостаток лидара заключается в более высокой стоимости и меньшей детализации текстур и цветов по сравнению с изображениями, полученными с помощью камер.

Средства сенсорной информации играют ключевую роль в системе управления манипулятором. Эти устройства формируют управляющие сигналы для манипулятора на основе целевых задач, заданных человеком-оператором, а также учитывают текущее состояние робота и окружающей среды.

С увеличением сложности робототехнических систем и условий их эксплуатации возникла необходимость в многомерном планировании движения. Например, рассмотрим персонального робота PR2 (см. рисунок 2), функционирующего в беспорядочной домашней обстановке, или гуманоидного робота Atlas (см. рисунок 3), который решает задачи навигации и манипуляции в неструктурированной среде. Для

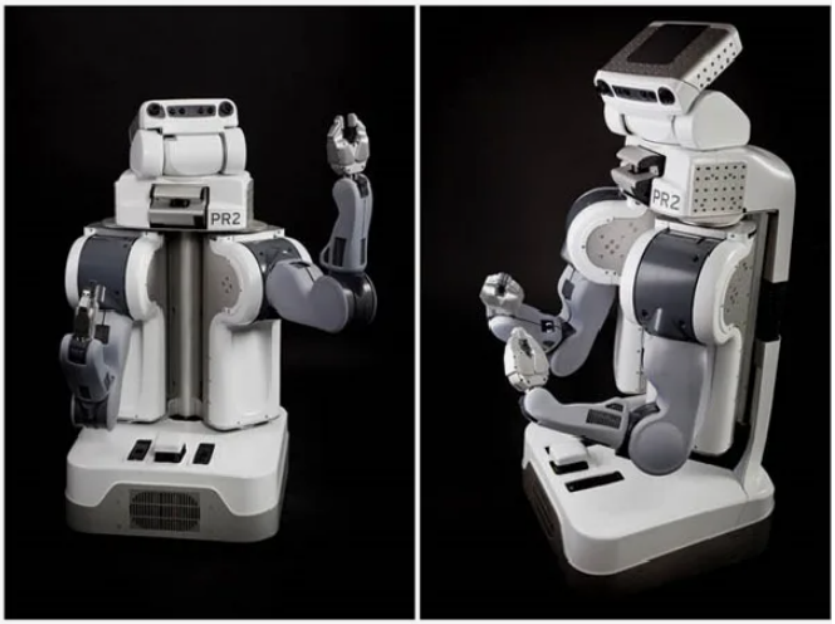


Рис. 2: Робот PR2

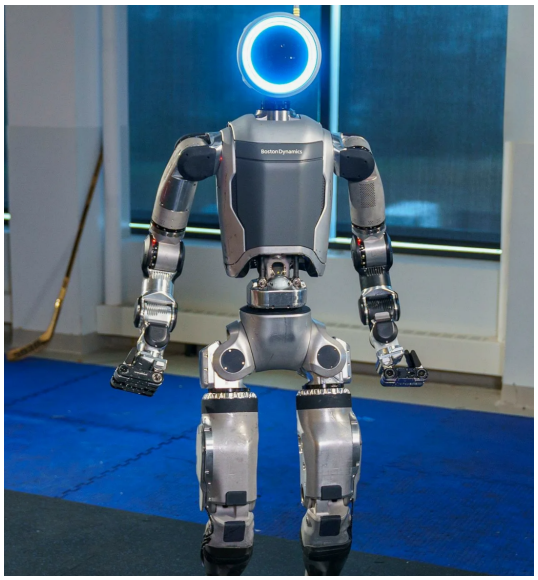


Рис. 3: Робот Atlas

эффективного выполнения этих задач, особенно учитывая необходимость избегания столкновений с препятствиями, критически важным становится планирование движения, что позволяет обеспечить высокопроизводительное выполнение заданий. В таких случаях особенно важным является время обработки данных, так как часто требуется перепланировка маршрута.

В рамках планирования движений для робототехнических систем выделяются две основные задачи: планирование маршрута (или пути) и планирование траектории. Для лучшего понимания этих задач следует определить ключевые термины:

- конфигурационное пространство — это совокупность всех обобщенных координат, которые точно описывают положение манипулятора в пространстве;

- рабочее пространство — это область, в пределах которой будет происходить разрешение задачи планирования движений.

Таким образом, для движения робота важно, чтобы конфигурационное пространство было свободно от препятствий.

Исходя из вышеупомянутых определений, можно сформулировать следующие понятия:

- планирование маршрута (пути) для робота — это процесс нахождения такой последовательности конфигураций, которая связывает начальную позицию с заданной конечной, обеспечивая при этом, что движение робота вдоль этого маршрута будет свободно от столкновений с препятствиями;

- планирование траектории движения робота — это процесс временной параметризации пути, то есть вычисление функций, зависящих от времени, для обобщенных координат, а также для скоростей и ускорений.

Таким образом, задача планирования траектории фокусируется на вычислении временных функций, которые определяют значения положений, скоростей и ускорений робота во время его перемещения по маршруту, описанному набором конфигураций. При этом существует множество различных подходов к осуществлению этого процесса, среди которых можно выделить классические методы и нейросетевые подходы.

Классические методы обобщают следующие характеристики:

- предсказуемость - эти методы обеспечивают предсказуемые результаты, что делает их подходящими для систем, где требуется стабильное и надежное выполнение задач;

- структурированность - классические методы часто основываются на четко определенных математических моделях и алгоритмах, что позволяет легко их анализировать и реализовывать;

- эффективность в статических средах - многие классические алгоритмы, такие как A* и алгоритм Дейкстры, хорошо работают в статических условиях, где

переменные не меняются в процессе выполнения. Это позволяет достигать оптимальных решений;

- ограниченная адаптивность - классические методы имеют ограничения в адаптации к динамическим условиям, так как они часто не учитывают изменения в окружении или движения других объектов.

Несмотря на это, классические методы могут эффективно обрабатывать ограничения на движение, такие как физические препятствия или предписания относительно безопасности. Тем не менее, они требуют дополнительных усилий для их интеграции в алгоритмы. Также стоит отметить, что они могут быть ограничены в скорости выполнения, особенно когда речь идет о сложных или многогранных пространственных задачах, что в свою очередь может стать проблемой при необходимости быстрого реагирования на изменение условий. Многие классические методы обеспечивают гладкие и непрерывные траектории, что особенно важно для робототехнических систем, чтобы избежать резких изменений в движении. К тому же, они часто основаны на простых численных решениях, что делает их доступными для реализации и понимания даже для тех, кто не имеет глубоких математических знаний.

Эти характеристики помогают понять, когда и как можно эффективно применять классические методы планирования траектории, а также выявляют их ограничения, что открывает возможности для разработки более современных и адаптивных подходов.

В отличие от классических, нейросетевые методы планирования траектории в робототехнике используют искусственные нейронные сети и демонстрируют превосходство в области:

- возможности обработки большого массива данных;
- генерации сложных траекторий;
- параллельной обработки данных;
- устойчивости к шуму.

Иными словами, нейросетевые подходы позволяют избежать ограничений классических методов и могут работать в недетерминированных средах столь же эффективно, как в детерминированных. Важно отметить, что нейросетевые модели могут учитывать широкий спектр факторов, влияющих на планирование траектории,

включая взаимодействие с объектами и людьми, что дает возможность для более точного прогнозирования. Используя нейросети, также возможно интегрировать информацию от различных сенсоров, таких как камеры, Lidar и другие, что позволяет создавать более полные и точные модели среды. Нейронные сети могут обучаться на изменяющихся данных, что позволяет роботам адаптироваться к новым условиям или неожиданным ситуациям в реальном времени. Это делает их особенно полезными в неструктурированных средах.

Таким образом, нейросетевые методы представляют собой мощный инструмент для решения задач планирования траектории, обеспечивая высокую адаптивность, эффективность и способность работать в сложных и изменяющихся условиях. Тем не менее, их применение требует большой вычислительной мощности и наличия качественных обучающих данных.

В конечном итоге, робот-манипулятор представляет собой сложную систему, состоящую из множества механизмов и технических устройств, что позволяет ему принимать положение в пространстве и оценивать обстановку вокруг за счет средств очувствления. Для перемещения робота-манипулятора используются алгоритмы, основанные на классическом и нейросетевом подходах. Детальное описание подходов, которые используются в современных условиях, будет представлено в следующем подразделе работы.

2.2 Обзор современного состояния проблемы

Планирование траектории является одной из важнейших задач в области робототехники. Эта задача включает в себя определение оптимального пути для перемещения роботов-манипуляторов с целью выполнения заданных задач. Однако современные средства планирования траектории имеют ряд проблем:

- высокая вычислительная сложность алгоритмов затрудняет быструю реакцию на изменения в динамических средах;
- недостаточная адаптация к неопределенности и изменчивости, поскольку многие алгоритмы основываются на статических условиях;
- низкий уровень навыков взаимодействия с объектами в среде, что может привести к неэффективному выполнению задач.

Эти проблемы требуют дальнейших исследований и разработки новых подходов.

Для планирования траектории роботов-манипуляторов используются разные алгоритмы:

- *алгоритмы, основанные на эвристическом поиске* (A^* [1], Dijkstra) применяются для поиска кратчайшего пути в графах и могут быть адаптированы для планирования траекторий, они эффективны для статических сред, но не подходят для динамических или сложных сцен;

- *алгоритмы на выборках* (RRT - Rapidly-exploring Random Tree [5] и PRM - Probabilistic Roadmap [4]) используют случайные выборки для построения траекторий в физическом пространстве, могут эффективно работать в высокоразмерных пространствах, что делает их популярными для многосвязных манипуляторов;

- *алгоритмы, основанные на оптимизации* (например, градиентном спуске - CHOMP [8] и численной оптимизации - TrajOpt [9]), рассчитывают параметры траектории, минимизируя целевую функцию, которая может включать в себя время, энергию или риск столкновения. Эти алгоритмы могут быть особенно полезны для задач, требующих непрерывной оптимизации;

- *методы на основе искусственных нейронных сетей* для планирования траекторий становится все более актуальными, сети могут обучаться на большом объеме данных и выявлять сложные паттерны, что облегчает адаптацию к изменяющимся условиям.

Выбор алгоритма для планирования траектории зависит от специфических требований задачи (скорость, точность и безопасность). Поскольку робот-манипулятор представляет собой сложную систему, имеющую несколько звеньев, алгоритмы для планирования его траектории можно условно разделить на 2 типа:

- те, которые могут точно рассчитать траекторию, но имеют высокую вычислительную сложность и не могут уложиться в адекватное время;

- те, которые работают быстро, но их результаты не могут быть выполнены на роботе безопасно.

Так, например, алгоритм RRTConnect [6] позволяет быстро находить траекторию, но может предложить путь, который физический робот не сможет выполнить. В таком случае существует риск, что робот перекрутится или столкнется с собственными звеньями. В отличие от RRTConnect, алгоритм TrajOpt предлагает траекторию, которую робот может почти гарантированно выполнить без коллизий, однако расчет занимает значительно больше времени. При работе с сложными сценами, наполненными препятствиями, TrajOpt может не сойтись за разумный временной интервал. Эффективность этого алгоритма может повыситься, если предоставить ему предварительно заданную начальную траекторию, которая избегает препятствий. Чем более оптимальной будет эта начальная траектория, тем выше вероятность успешного завершения планирования. В этой связи для решения задачи планирования может быть использована комбинация алгоритмов RRTConnect и TrajOpt: RRTConnect рассчитывает начальное приближение траектории, которое потом оптимизирует TrajOpt. Такая связка работает эффективнее, чем использование каждого из алгоритмов по отдельности.

Алгоритмом для проведения исследования выбран TrajOpt, поскольку он позволяет находить безопасное решение для робота-манипулятора. TrajOpt применяет метод штрафов, чтобы обрабатывать столкновения. Он гарантирует непрерывную безопасность во времени, что позволяет избежать столкновений на протяжении всей траектории.

TrajOpt - это алгоритм планирования траектории для решения задач движения роботов. Он основан на методе оптимизации, который пытается найти траекторию, удовлетворяющую ограничениям на столкновения. Алгоритм позволяет роботу находить кратчайший путь к цели и обходить препятствия. TrajOpt в отличие от CHOMP оптимизирует траекторию, которая изначально проходит через препятствия. Используя метод штрафов, алгоритм находит способ избежать столкновения робота с препятствиями. Чтобы рассчитать штраф, требуется установить факт столкновения. Если происходит столкновение, вычисляется расстояние, на которое эти объекты (звено робота и препятствие) должны оттолкнуться друг от друга. Для проверки факта столкновения и глубины проникновения объектов используются алгоритм алгоритм Гилберта-Джонсона-Кирти (GJK) [2] и алгоритм расширяющегося многогранника [7] соответственно.

При оптимизации траектории с помощью TrajOpt, алгоритм стремится избежать столкновений на каждом временном шаге траектории. Робот и препятствия рассмат-

риваются как твердые тела, которые не должны сталкиваться и пересекаться.Если есть пересечение, TrajOpt определяет, насколько сильно фигуры проникают друг в друга и назначает штраф, который при изменении глубины проникновения на следующих этапах расчета траектории будет постепенно уменьшаться, пока не станет близким к нулю.

Пусть A и B - объекты (твердые тела), T - расстояние между объектами и sd – знак расстояния. Если расстояние положительно, то два объекта не сталкиваются. Если расстояние трицательно, объекты A и B сталкиваются, и необходимо найти безопасное расстояние (равное или большее T), на которое можно оттолкнуть A от B, чтобы вывести объекты из состояния столкновения. T в этом случае называется глубиной проникновения.

Метод компенсации за столкновение может быть описан следующим образом:

$$sd_{AB}(\theta) \approx \widehat{n}(P_B - P_A(\theta)),$$

где sd расстояние со знаком, θ – временной шаг, $\widehat{n}$ вектор нормали, P_A and P_B – точки соприкосновения объектов. Расстояние между двумя объектами $A, B \subset R^3$ отличное от нуля для непересекающихся множеств, определяется как:

$$dist(A, B) = inf\{\|T\| \mid (T + A) \cap B \neq \varnothing\}$$

Аналогично определяется глубина проникновения, если она не равна нулю, отличная от нуля для перекрывающихся форм:

$$penetration(A, B) = inf\{\|T\| \mid (T + A) \cap B = \varnothing\}$$

Расстояние со знаком определяется следующим образом:

$$sd(A, B) = dist(A, B) - penetration(A, B)$$

На рисунках 4 и 5 представлен принцип расчета штрафа за столкновение.

Описание основных понятий на рисунках:

- A and B - объекты,
- T - дистанция между объектами,
- sd - расстояние со знаком,

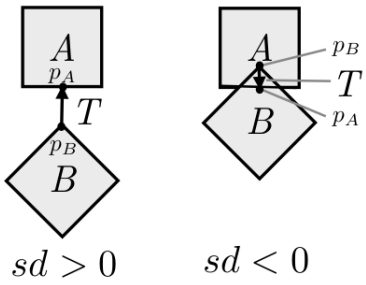


Рис. 4: Минимальное расстояние

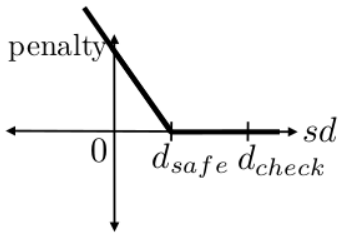


Рис. 5: Штраф за столкновение

- θ – временной шаг,
- $\widehat{n}$ - нормаль вектора,
- P_A and P_B – точки контакта между объектами.

Чтобы рассчитать штраф за столкновение, используются алгоритмы:

- алгоритм Гилберта-Джонсона-Кирти (GJK) для определения факта столкновения между телами [2];
- алгоритм расширяющегося многогранника (EPA) для определения глубины проникновения [7].

Алгоритм Гилберта-Джонсона-Кирти (GJK). Алгоритм используется для вычисления расстояния между объектами и выявляет факт столкновения (коллизии). Алгоритм GJK используется для определения конфигурационного пространства препятствий. Для пары выпуклых оболочек (A и B) их конфигурационного пространства расстояние будет равно:

$$d(A, B) = min\{\|x\| : x \in A - B\}$$

Объекты A и B пересекаются, если их конфигурационное пространство препятствий содержит начало координат системы координат:

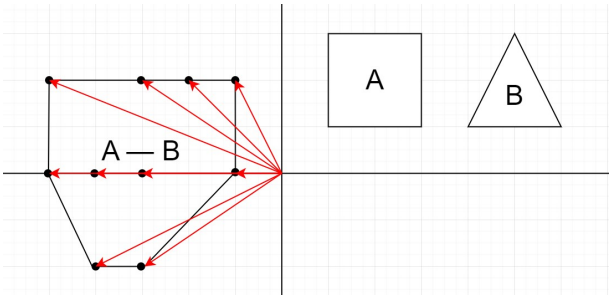


Рис. 6: Фигуры не пересекаются

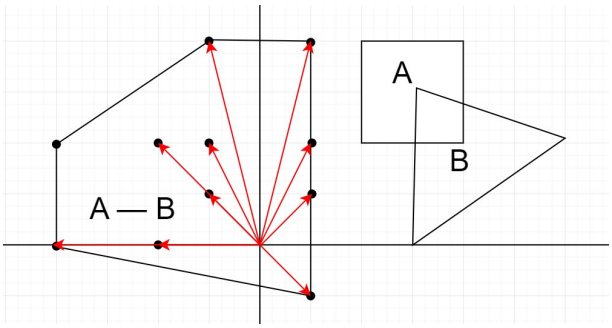


Рис. 7: Фигуры пересекаются

$$A \cap B \neq \emptyset \equiv 0 \in A - B$$

GJK использует разность Минковского [3]. Этот алгоритм вычисляет как разность координат фигур A и B. Разность Минковского множеств A и B является максимальным множеством C таким, что $C + B \subset A$. Каждая вершина каждой фигуры вычитается из каждой вершины другой фигуры. Координаты этой разности множеств образуют многогранник C.

Чтобы найти разницу Минковского, необходимо:

- попарно вычесть вершины одной фигуры из вершин другой;
- получить набор векторов из начала координат;
- соединить их в выпуклую оболочку.

Многоугольник $A - B$ - это разность Минковского. Если разность Минковского не содержит начала координат, то фигуры A и B не пересекаются (рис. 6). Если разность Минковского содержит начало координат, то фигуры A и B пересекаются (рис. 7).

Вычисление разности Минковского может занять много времени, особенно если вы проверяете пересечение двух фигур, состоящих из множества точек. Сложность

вычисления $A - B$ равна $O(n^2)$, потому что вам нужно выполнить итерацию по всем вершинам каждой фигуры. Однако нет необходимости вычислять полную разницу. Следовательно, алгоритм GJK использует понятия разности Минковского и оптимизирует ее. Цель алгоритма GJK состоит в том, чтобы определить, находится ли начало координат в пределах разности Минковского за минимальное число итераций. Необходимо определить самые дальние вершины в определенном направлении, чтобы получить увеличение скорости. Поиск самой дальней вершины выполняется путем перебора набора вершин и нахождения вершины с наибольшим скалярным произведением в направлении от начала координат. Необходимо, чтобы наименьшая крайняя вершина из B была вычтена из самой крайней вершины из A. Эти вершины называются опорными точками и дают представление о разнице Минковского без вычисления общей разницы. Чтобы определить, охватывает ли разность Минковского начало координат, нужно создать симплекс вокруг начала координат или убедиться, что это невозможно. Для двумерного пространства это треугольник. Для трехмерного - тетраэдр. Поиск в направлении начала координат позволяет алгоритму быстро сходиться. Нужно начать с произвольной вершины, затем добавлять или удалять вершины на каждой итерации, пока не будет окружено начало координат или не станет ясно, что это невозможно. Если начало координат находится внутри симплекса, то произошло столкновение.

Алгоритм GJK позволяет ответить на вопрос, существует ли столкновение между объектами. Алгоритм EPA поможет ответить на вопрос о том, как далеко объекты A и B нужно отодвинуть друг от друга, чтобы избежать столкновения.

Алгоритм расширяющегося многогранника (EPA). Алгоритм EPA берет последний симплекс, содержащий начало координат (который был получен во время реализации алгоритма GJK), и находит нормаль столкновения, то есть кратчайший вектор, чтобы оттолкнуть фигуры друг от друга. EPA называется алгоритмом расширяющегося многогранника, потому что он расширяет симплекс, добавляя к нему вершины, пока не найдется кратчайшая нормаль от грани, которая расположена на исходном многоугольнике. Проще говоря, симплекс становится многогранником после того, как начать добавлять к нему дополнительные вершины.

Анализ исследовательских статей показал, что TrajOpt:

- чувствителен к начальному приближению;
- имеет высокую вычислительную сложность;

- может застревать в локальных минимумах;
- плохо подходит для использования в быстро меняющихся средах из-за медленной скорости расчетов.

Но несмотря на недостатки, алгоритм TrajOpt активно используется для формирования траектории движения роботов-манипуляторов и может давать гарантированный результат успешного выполнения предложенной траектории. Этот алгоритм предлагает траектории, которые робот может выполнить безопасно для себя и окружающих. В нем нет вероятностной составляющей. Именно поэтому алгоритм TrajOpt был выбран в качестве исследуемого метода в данной работе.

Цель работы - предложить подход который генерирует лучшее начальное приближение траектории для последующей оптимизации в TrajOpt.

Недавние исследования показывают, что ученые часто обращаются к алгоритму TrajOpt для планирования траектории робота-манипулятора. Авторы обзора [32] рассматривают методы генерации движений, разработанные в робототехнике за последние 50 лет. Они выделяют как традиционные подходы на основе явных моделей, так и современные методы с использованием имплицитных моделей, обучающихся на данных. Будущее традиционных подходов, по их мнению, заключается в адаптации и обогащении новыми данными, что приведет к более тесной интеграции методов. Это подчеркивает значительные перспективы для дальнейших исследований адаптации TrajOpt.

В статье [30] представлены нововведения, которые могут улучшить планирование и управление коллаборативными роботами. Авторы предлагают адаптировать алгоритмы, такие как TrajOpt и CHOMP, с учетом динамических изменений окружения, включая предсказание поведения человека и оптимизацию взаимодействий с использованием методов обучения с подкреплением. Обсуждается также использование фреймворков для предсказания человеческих намерений, что позволяет роботам корректировать свои действия.

Среди проблем, связанных с TrajOpt, можно отметить зависимость от начальных условий и трудности в решении комплексных ограничений ([36] и [37]). Методы, подобные TrajOpt, могут застревать в локальных минимумах и не всегда подходят для длинных или сложных траекторий. Традиционные методы, такие как Grad TrajOpt и MPPI, также испытывают сложности в исследовании пространства действий и получении полезной информации о задаче [31].

Новые подходы, сравниваемые с TrajOpt

Новые методы направлены на улучшение планирования движений роботов, учитывают динамические среды и сложности, с которыми сталкивается TrajOpt.

Авторы [11] предлагают метод, расширяющий планировщики движения на основе выпуклой программы (SCP). Он учитывает неопределенности в сенсировании. Описанная методика ROMP реорганизует подзадачи в надежные оптимизационные задачи, обеспечивая безопасные траектории для роботов с высокой степенью свободы (DOF) и включает наблюдение за ограничениями для предотвращения столкновений, предложенными в TrajOpt.

В работе [12] представлен алгоритм iAGP-STO (инкрементальный стохастический оптимизационный алгоритм на основе адаптивного гауссового процесса), который сочетает численные методы, такие как ускоренный метод градиентного спуска (AGD), со стохастическими методами для улучшения планирования траекторий. При сравнении с TrajOpt и другими методами, iAGP-STO демонстрирует большую успешность и эффективность в определенных условиях благодаря адаптации к динамически меняющимся ситуациям. Похожая методика представлена авторами [16], которые разработали адаптивную гауссовую процессную стохастическую оптимизацию траектории (AGP-STO). Этот метод также комбинирует численную и стохастическую оптимизацию, улучшая эффективность и надежность в условиях высокой сложности и наличия препятствий.

В алгоритме PINSAT [34] оптимизация траектории происходит в два этапа: сначала создается краткосрочная траектория, затем проводится её улучшение с использованием «теплого старта» (warm starting) для более длинных траекторий. Это позволяет значительно ускорить процесс оптимизации. TrajOpt рассматривается в контексте кинодинамического планирования, где акцент делается на обеспечение динамической совместимости и удовлетворение ограничений при планировании. Алгоритм PINSAT показывает улучшение по сравнению с традиционными методами, предлагая более эффективные способы для выполнения кинодинамических задач.

TrajOpt упоминается как один из методов оптимизации [39], который использует последовательное квадратичное программирование для решения задач, связанных с планированием траекторий. Хотя он продемонстрировал свою эффективность, новый подход авторов предлагает более мощные возможности благодаря использованию облаков точек для достижения целей и избегания столкновений, а также за счет интеграции планирования захвата в процесс оптимизации движения.

В исследовании [29] предложен планировщик траекторий удаляющегося горизонта, который использует комбинацию нового представления множества достижимых объектов и функции точного расстояния со знаком для создания доказуемо безопасных планов движения. Во время выполнения алгоритм использует параметризованные траектории для вычисления наборов достижимых объектов, которые завышенно аппроксимируют объем движения робота, и можно гарантированно выбрать безопасную траекторию без столкновений.

В работе [14] представлен метод планирования, использующий конечные ряды Фурье для представления непрерывного движения. Этот подход регулирует гармоники движения суставов манипулятора, обеспечивая плавность и точное отображение траектории в M-мерном конфигурационном пространстве. Также метод включает поля столкновений для обхода препятствий.

Авторы исследования [27] представляют новый алгоритм под названием Motion Planning Diffusion (MPD), который направлен на улучшение планирования траекторий для автономных роботов. TrajOpt рассматривается как один из популярных методов, ориентированных на локальную оптимизацию траекторий и минимизацию функции стоимости, например, избегание столкновений. В MPD отмечается, что такие методы, как TrajOpt, могут застревать в локальных минимумах, особенно когда начальная траектория определяется как прямая линия между стартовой и целевой конфигурацией. Это подчеркивает важность более продвинутых подходов, таких как MPD, которые учатся на предыдущих решениях и могут успешно работать в сложных средах.

TrajOpt является одним из методов оптимизации траекторий, используя последовательное квадратичное программирование (SQP), но страдает от вычислительной сложности, что ограничивает его применение. В отличие от него, новый подход авторов [40] на основе метода AL-iLQR демонстрирует более эффективное и гибкое планирование пути, обеспечивая быструю сходимость и высокую стабильность, что делает его более быстрым и эффективным в условиях сложных препятствий.

Новые методы планирования движений роботов, анализируемые в работе, предлагают улучшенные решения для работы в динамических средах и сложных условиях, что позволяет повысить эффективность и безопасность планирования траекторий по сравнению с TrajOpt.

Улучшения алгоритма TrajOpt

Последние работы демонстрируют множество методов, направленных на улуч-

шение алгоритма TrajOpt для планирования траекторий роботов.

Авторы [15] предлагают интеграцию выбора различных поз захвата в процесс оптимизации траекторий. Этот метод позволяет роботу исследовать несколько поз для достижения более эффективной траектории при выполнении манипуляций, особенно в сложных и загроможденных средах. Постепенное сокращение горизонта оптимизации к целевой области повышает вычислительную эффективность и способствует получению более плавных траекторий, помогая избежать сложных конфигураций и повышая успех выполнения задач. В статье TrajOpt рассматривается как черный ящик для оптимизации траекторий. Авторы описывают процесс, в котором TrajOpt принимает на вход начальные и конечные позы и генерирует оптимальные траектории. При этом новые позы захвата выбираются на каждом шаге алгоритма, а TrajOpt использует их для получения решения.

Хотя TrajOpt считается значительным прорывом в оптимизации траекторий, в работе [19] представлен алгоритм STODI, который выделяется благодаря своему подходу к случайной оптимизации и способности обрабатывать более сложные задачи без необходимости в явной информации о градиенте. Это делает его более эффективным для сложных целей. STODI не только опровергает ограничения, связанные с TrajOpt, но и предлагает значительные улучшения в области имитации и оптимизации движений роботов.

Метод SPLANNING [28] улучшает реализованные в TrajOpt подходы, используя более жесткие ограничения по избежанию столкновений и работая с непрерывными представлениями вместо дискретной геометрии. Это предоставляет большую гибкость в планировании движения робота.

Также стоит отметить работу [20], в которой представлен новый подход, комбинирующий графы выпуклых множеств (GCS) с нелинейной оптимизацией для решения задач некомплексной траектории. Этот метод позволяет более эффективно обходить локальные минимумы и повышает качество решения. В документе упоминаются различные методы оптимизации траекторий, включая GCSTrajOpt, который имеет ограничения по управлению скоростью и приводит к неприемлемым ускорениям. Напротив, NGCSTrajOpt, предлагаемый авторами, позволяет контролировать как скорость, так и ускорение, что обеспечивает динамически выполнимые решения без необходимости последующей обработки.

Наконец, в работе [43] упоминается метод TrajOpt как пример оптимизации траектории, который непосредственно оптимизирует движение в пространстве суставов

с учетом ограничений. Однако авторы указывают на проблемы традиционных методов, такие как чувствительность к начальной траектории и вероятность застревания в локальных минимумах. Таким образом, представленный фреймворк предлагает альтернативу, снижая вычислительные затраты и улучшая общий процесс оптимизации по сравнению с TrajOpt.

Новые подходы, направленные на улучшение TrajOpt, повышают безопасность и адаптивность движений, уменьшая вероятность застревания в локальных минимумах, и учитывают более широкий спектр параметров, в том числе скорость и ускорение.

Улучшения, основанные на Machine learning

Методы машинного обучения помогают решить проблемы, с которыми сталкивается TrajOpt, путем внедрения адаптивных стратегий, способных учиться на предыдущих опытах и подстраиваться под уникальные условия среды. Кроме того, нейросетевые подходы могут улучшить инициализацию начальных траекторий, обеспечивая более качественные стартовые состояния для оптимизации и, тем самым, способствуя более быстрому и эффективному поиску решений.

В контексте решения проблем, связанных с TrajOpt, используются несколько подходов машинного обучения (ML).

1. Обучение с подкреплением (Reinforcement Learning, RL) позволяет моделям обучаться на основе взаимодействия с окружающей средой, что помогает им находить более оптимальные траектории для захвата или манипуляций, избегая застревания в локальных минимумах.

В работе [22] обсуждается метод Soft Analytic Policy Optimization (SAPO), который включает элементы обучения с подкреплением для манипуляции с деформируемыми объектами в робототехнике. Метод TrajOpt используется для оптимизации действий с целью максимизации вознаграждений.

Метод, который сочетает RL и оптимизацию траектории для для управления компромиссами между безопасностью и вознаграждениями, предлагают авторы [23]. Система RL генерирует действия, которые преобразуются в безопасные траектории с помощью оптимизатора (TrajOpt), улучшая безопасность операций и позволяя агенту обучаться в динамичных условиях.

Авторы статьи [35] представляют ITPNet — глубокую сеть для предсказания начальных траекторий, улучшающую работу оптимизационных планировщиков в робототехнике. Качество начальной траектории критически важно для эффективности

планирования. ITPNet базируется на предыдущих методах и предсказывает траектории для «разогрева» оптимизаторов. Алгоритм значительно улучшает предсказание начальных траекторий, повышая эффективность и надежность планирования. Аналогично, в работе [17] начальная траектория, которая затем подается в TrajOpt, генерируется с помощью обучения с подкреплением.

Авторы [42] предлагают метод Reflect-then-Plan (RefPlan), который представляет собой инновационный подход к моделям обучения с подкреплением (RL), обращающим внимание на непредсказуемость в условиях ограниченных данных. TrajOpt упоминается как часть алгоритма, использующего модель предсказательного управления (MPC), которая задействует TrajOpt для оптимизации траекторий в условиях неопределенности. С помощью этого метода агенты могут генерировать оптимальные планы действий, опираясь на изученные динамические модели и ранее накопленные знания о среде.

Авторы [25] предлагают новый оффлайн алгоритм глубокого имитационного обучения для оптимального планирования траекторий в нестабильных динамических системах, имитирующий итеративный линейный квадратичный регулятор (iLQR) для ускорения вычислений. В документе упоминается TrajOpt как существующий метод оптимизации для генерации безколлизийных траекторий, но новая методология сфокусирована на нестабильных системах и оптимизации начальных условий, что расширяет её применение.

Исследователи [10] предложили фреймворк, в котором рассматривается потенциально неизвестная среда и применяется обучение с обратным подкреплением (IRL) для планирования движения с помощью TrajOpt.

2. Нейронные сети (сверточные и рекуррентные сети), позволяют моделировать сложные зависимости и динамику объектов, что помогает улучшить качество инициализации траекторий и предсказания движений.

В методе LeTO [41] используется нейронная сеть CNN+LSTM для обучения на основе демонстраций человека. TrajOpt использует последовательную выпуклую оптимизацию для генерации плавных и безопасных траекторий, но LeTO обеспечивает более высокую адаптивность к визуомоторным политикам и лучшее выполнение в реальных задачах. LeTO показывает способность генерировать плавные траектории с учетом ограничений как в симуляциях, так и на реальных роботах, демонстрируя робастность к аномальным данным.

Также существует подход, улучшающий TrajOpt, который использует графовые

нейронные сети (GNN) для оценки расстояния до столкновения [18]. Он применяется для планирования в условиях, когда цель окружена препятствиями (например, предмет в приоткрытой тумбочке стола).

3. Генеративные модели, такие как диффузионные или вариационные автокодировщики, могут быть использованы для создания большого количества разнообразных стартовых траекторий, повышая вероятность нахождения оптимального пути.

В работе [33] упоминается о роли диффузионных моделей для генерации начальных траекторий, что позволяет улучшить процесс планирования движения. TrajOpt играет ключевую роль в работе алгоритма: после генерации начальных траекторий с помощью диффузионной модели PRESTO применяет оптимизацию на основе TrajOpt, обеспечивая гладкость и избегание столкновений. Штрафы TrajOpt используются для обучения диффузионной модели, что позволяет генерировать более качественные и безопасные траектории.

Авторы [24] предлагают метод управления с помощью диффузионных моделей, названный Diffusion Model Predictive Control (D-MPC), который превосходит TrajOpt благодаря более простому планировщику SSR и избегает сложных методов оптимизации траекторий, достигая лучших результатов в задачах управления. В то же время, CROWS (Conformalized Reachable Sets for Obstacle Avoidance With Spheres) [38] представляет новую методологию для планирования движений роботов, обеспечивая безопасные и эффективные реализационные планы в реальном времени. Используя нейронные сети и минимизируя количество эвристик, CROWS достигает конкурентоспособных результатов по сравнению с TrajOpt, оставаясь при этом свободным от столкновений.

4. Обработка естественного языка (NLP) с TrajOpt улучшает взаимодействие пользователя с системой, позволяя более точно интерпретировать команды, адаптивно управлять траекториями и учитывать контекстные факторы при оптимизации манипуляций.

TrajOpt применяется для задач захвата объектов [26] с управлением через естественный язык. Он требует непрерывной дифференцируемой эталонной траектории и использует тройную интеграторную динамику для плавности движений, что увеличивает надежность и точность манипуляций. Модель MP-TrajOpt оптимизирует пути без заранее заданной траектории, фокусируясь на определении безколлизийных траекторий, что критично для манипуляций.

В работе [13] представлен метод обхода препятствий с использованием изобра-

жения с камеры, текстовой подсказки и конфигурации робота. Алгоритм TrajOpt применяется для проверки столкновений, а текстовые подсказки информируют о препятствиях. Планирование ограничивается языком, и для получения информации о среде используется лишь одна камера.

Методы машинного обучения значительно улучшают функциональность TrajOpt, позволяя адаптивно подстраиваться под уникальные условия среды и повышая качество инициализации начальных траекторий. Использование обучения с подкреплением и нейронных сетей способствует созданию более оптимальных и безопасных планов движения.

Таким образом, TrajOpt хоть и является алгоритмом, который может найти гарантированно выполнимый роботом путь, нуждается в улучшении и усовершенствовании. Он крайне чувствителен к начальной траектории, и эту траекторию ему необходимо подать на вход. Только так можно добиться эффективной работы алгоритма TrajOpt. Интеграция нейронных сетей может повысить эффективность, надежность и безопасность планирования траекторий, что, в свою очередь, сделает робототехнические системы более адаптивными к меняющимся условиям.

2.3 Формальная постановка задачи

Формализовать задачу можно следующим образом. У нас есть текущее положение манипулятора, целевое положение манипулятора, данные с сенсоров (камеры, лидар).

Необходимо:

1. **или** построить и выполнить траекторию T, которая будет проходить в свободном от препятствий пространстве;

2. **или** установить, что выполнение такой траектории невозможно;

3. **или** во время выполнения после изменения среды установить, что выполнение траектории невозможно.

Для доработки представления препятствий потребуется:

1. отфильтровать данные с сенсоров, которые находятся за пределами достижимости робота-манипулятора;

2. преобразовать Pointcloud в Octomap;

3. объединить кубы в меши-параллелепипеды (объединение постролить от начального приближения траектории).

Общая формула:

$$\min_T F(T) = \begin{cases} T(0) = q_{\text{start}}, \\ T(1) = \text{N}(q_{\text{start}}, q_{\text{goal}}), \\ \forall t \in [0,1], T(t) \in C_{\text{free}} \setminus M, \\ S(T(t)) = 0 \end{cases}$$

где

- C - конфигурационное пространство манипулятора (C_{free} - без препятствий, C_{obs} - с препятствиями, так что $C = C\text{free} \cup C_{\text{obs}}$);
- q_{start} - начальное положение манипулятора в конфигурационном пространстве;
- q_{init} - начальное приближение траектории манирулятора в конфигурационном пространстве;
- q_{goal} - целевое положение манипулятора в конфигурационном пространстве;
- N - нейросеть, принимающая q_{start} и q_{goal} и выдающая q_{init};
- M - представление препятствий,
  $M = \text{агрегация}_{Z,Y,X}\ (\{\text{куб}(p, s(\|p - p_{\text{init}}\|)) \mid p \in \{p \in C \mid I(p_x, p_y, p_z) = 1\}\})$, $s(d) = s_{\min} + k(d - d_{\min})$, $s_{\min}$ - минимальный размер куба, k - коэффициент увеличения, $d_{\min}$ - минимальное расстояние для $s_{\min}$, $I(x, y, z) = 1$,
  если точка (x, y, z) в зоне достижимости и 0 - иначе;
- S - совокупность данных с сенсоров, которые м.б. представлены как отображение $S : C \to 0, 1$, где $S(q) = 0$ означает, что конфигурация свободна, а $S(q) = 1$ означает, что конфигурация недоступна из-за препятствия;
- $T : [0, 1] \to C$ - траектория, которая является непрерывной функцией в конфигурационном пространстве.

Таким образом, основные аспекты задачи планирования траектории (а именно построение и выполнение безопасной траектории T, свободной от препятствий) для манипуляторов включают текущее положение манипулятора, целевое положение и

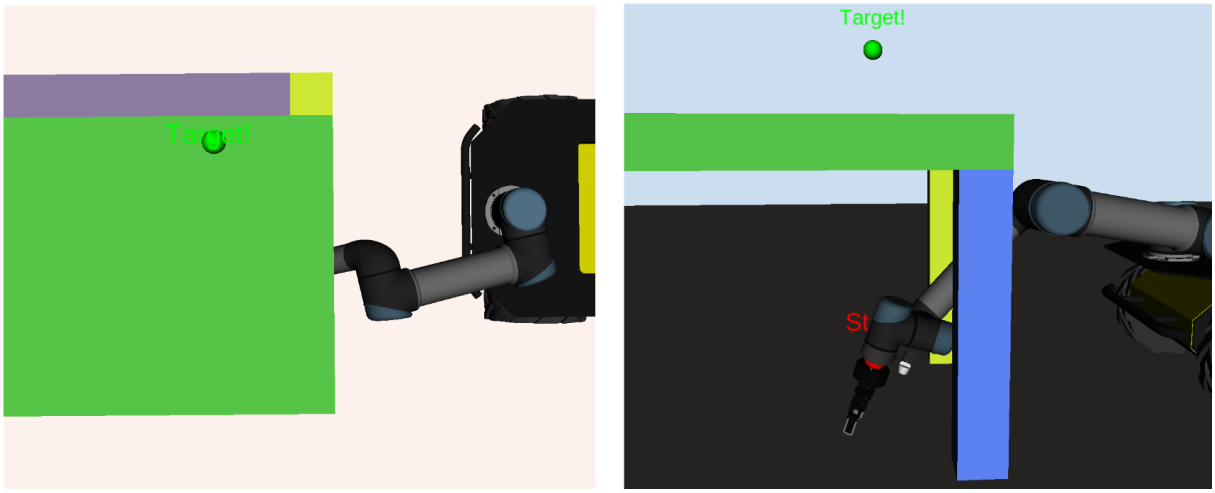


Рис. 8: Сложная сцена для планирования траектории

информацию с сенсоров. Для достижения поставленных целей требуется обработка данных с сенсоров, включая фильтрацию, преобразование Pointcloud в Octomap и объединение кубов в меши-параллелепипеды.

В текущем разделе описаны ключевые концепции, связанные с манипуляторами и их кинематикой, что является основой для дальнейших исследований в области планирования траекторий. Обзор современных технологий и методов демонстрирует значительное развитие в робототехнике и искусственном интеллекте. Рассмотренные алгоритмы и подходы, включая A*, Dijkstra, TrajOpt, а также нейросетевые методы и обучение с подкреплением, подчеркивают эффективность и разнообразие стратегий планирования движений. Понимание сложных задач требует наличия адаптивных алгоритмов, что становится важным аспектом разработки современных роботизированных систем.

3 Основной раздел работы

TrajOpt является алгоритмом, который позволяет находить выполнимые траектории. Однако сам по себе между двумя положениями робота (начальным и целевым) он чаще всего не может найти путь, если предполагаемая траектория должна огибать препятствие. Например, построить траекторию в таких условиях, как на рисунке 8 у него не получилось.

Поэтому было выдвинуто 2 основных предположения, как улучшить вероятность успеха работы алгоритма:

- предложить свой вариант представления препятствий;
- изучить работу алгоритма и найти варианты, как его доработать.

Предлагаются следующие улучшения алгоритма:

- подобрать, доработать или переопределить параметры внутри алгоритма TrajOpt, чтобы улучшить результат работы алгоритма для планирования движения робота-манипулятора в среде с препятствиями (повысить успех расчетов, сократить время планирования);
- разработать метод поиска начальной траектории без пересечения с препятствиями;
- доработать представление препятствий так, чтобы найти баланс между производительностью алгоритма и скоростью обработки данных с сенсоров робота.

3.1 Базовое решение

В качестве базового решения для планирования использовались алгоритмы RRTConnect и TrajOpt. Реализация RRTConnect взята из фреймворка MoveIt (см. рисунок 9). Известно, что успех алгоритма TrajOpt зависит от начальной траектории. В данном случае алгоритм RRTConnect предоставляет начальную траекторию, а TrajOpt её оптимизирует. Если траектория для инициализации не задана, то берется прямая линия от начальной до целевой позиции манипулятора и интерполируется по всей длине. Для каждого эксперимента одна траектория вычисляется в MoveIt, а затем оптимизируется один раз с использованием TrajOpt.

На практике не все траектории, которые строит RRTConnect, TrajOpt может оптимизировать. На рисунке 10 показаны траектории, с которыми Trajopt не справился, а на рисунке 11, наоборот, приведены удачные решения.

Для оценки базового решения и предлагаемых методов использовался валидационный датасет (подробнее о том, как он собирался, описано в разделе о проведении экспериментов). Этот датасет включает в себя положения роботов в виде джоинтов, и гарантируется, что все положения достижимы, для них есть решения обратной кинематики и TrajOpt на сцене без препятствий между предложенными положениями может построить траекторию. Процент успеха при валидации на этом датасете

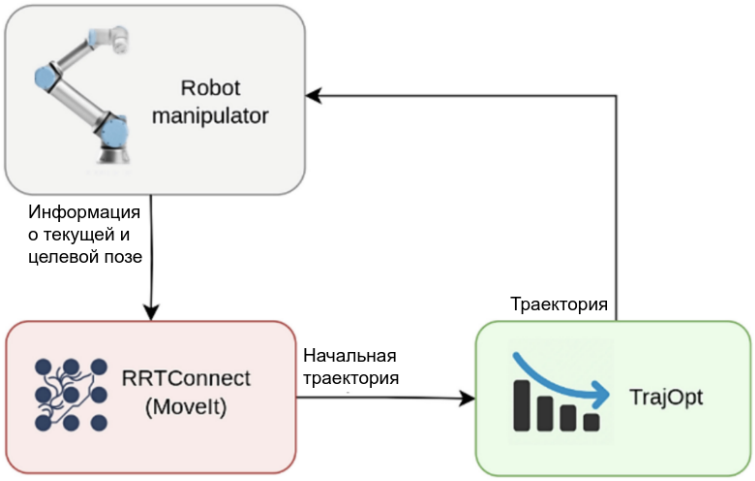


Рис. 9: Инструменты для планирования и оптимизации траектории

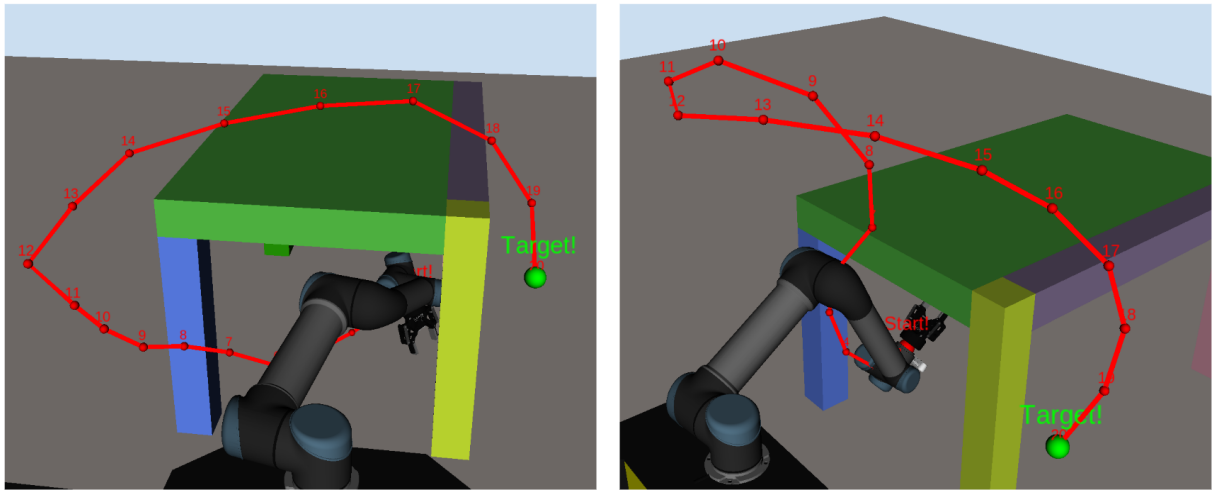


Рис. 10: Траектория RRTConnect

у RRTConnect составляет 65%. То есть только 65% траекторий, которые предложил RRTConnect, алгоритм TrajOpt смог оптимизировать.

Что касается представления препятствий на сцене, то в качестве базового решения использовался Octomap. И TrajOpt, и RRTConnect умеют обрабатывать препятствия в этом виде.

3.2 Предлагаемый метод

Предлагается метод, который будет устранять недочеты алгоритма TrajOpt, тем самым повышая эффективность его планирования. Итак, главные недочеты - это:

- чувствительность к начальной траектории (если траектория проходит сквозь препятствия, то TrajOpt хуже справляется с планированием, и чем дальше

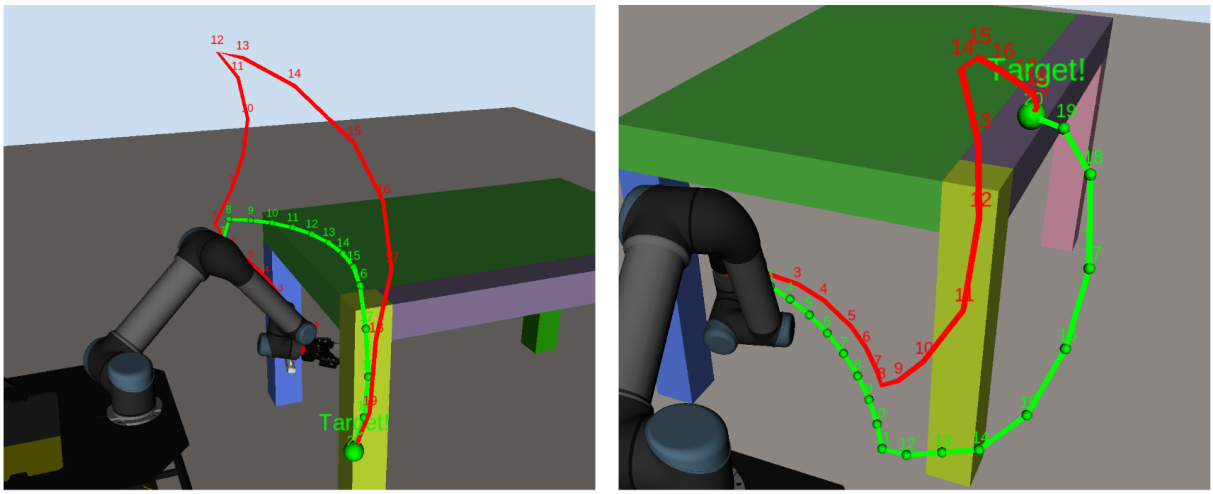


Рис. 11: Траектория RRTConnect (красным) и TrajOpt (зеленым)

нужно эту траекторию оттолкнуть от препятствия, тем выше шанс неудачи);

- высокая вычислительная сложность т.к. алгоритм итеративно рассчитывает факт столкновения с препятствием и глубину проникновения в препятствие на каждом шаге своего исполнения, то если на сцене расположить много небольших объектов, то TrajOpt может не успеть произвести расчеты за адекватное время.

Предлагаемый метод направлен на:

- формирование начального приближения траектории (цель - сформировать такую траекторию, которая позволит за оптимальное время алгоритму TrajOpt найти решение);
- преобразование представления препятствий (укрупнение препятствий на сцене с целью снизить вычислительную сложность алгоритма).

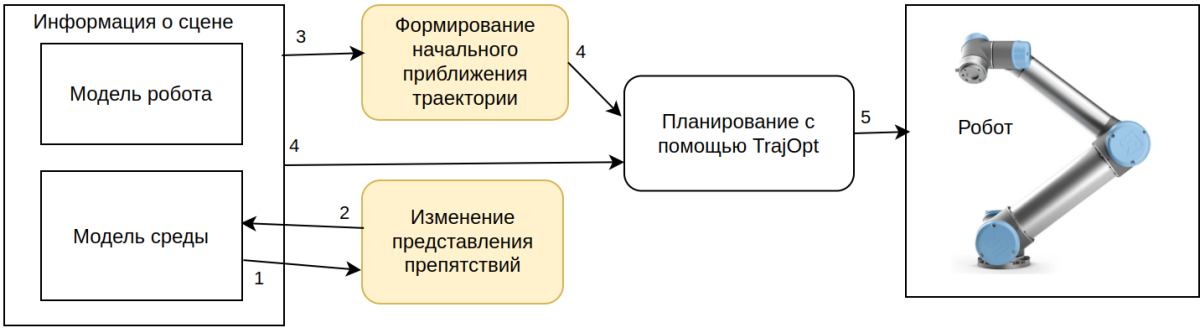


Рис. 12: Предлагаемый метод

На рисунке 12 представлена схема работы метода:

1. берется модель среды (объекты, препятствия), которая представлена в виде Octomap, и преобразуется в параллелепипеды;
2. укрупненные препятствия отображаются на сцене;
3. на основании информации о сцене (модель робота и модель среды) формируется начальное приближение траектории;
4. информация о сцене с моделью робота и укрупненными препятствиями и начальное приближение траектории подается в TrajOpt;
5. TrajOpt оптимизирует траекторию и в случае нахождения успешного решения, подает на робота.

Программная реализация представлена на рисунке 13. Участки, отмеченные желтым, были написаны в рамках проведения научной работы.

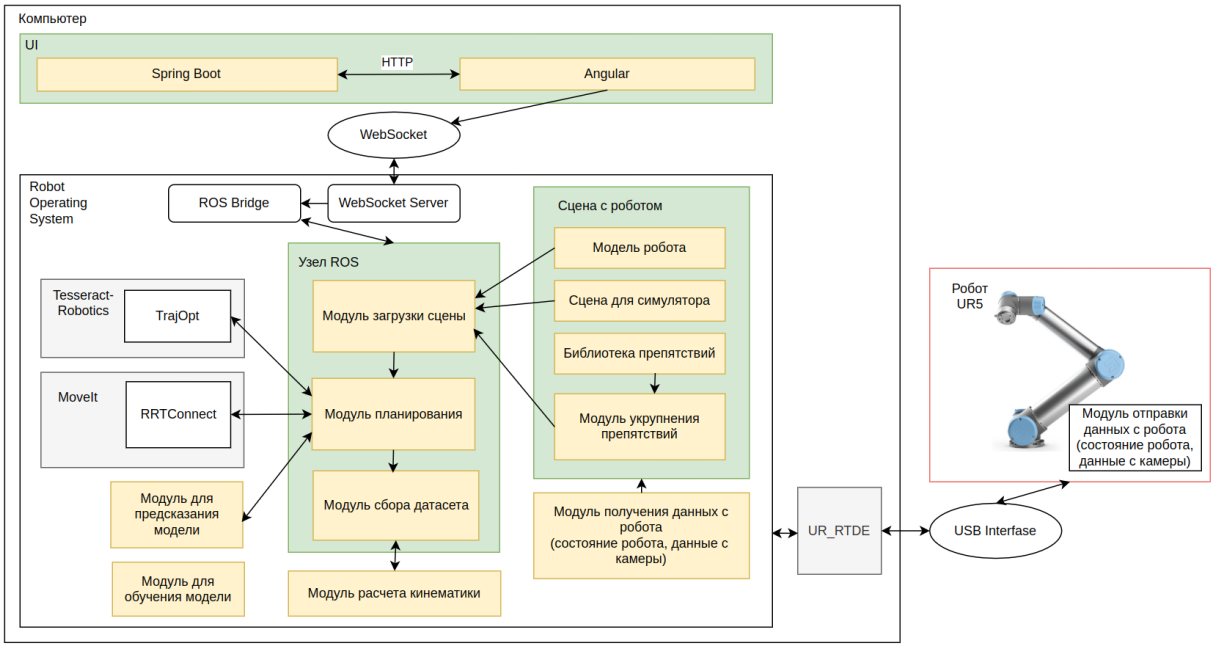


Рис. 13: Программная реализация

Основная часть программного кода написана для Robot Operating System (ROS) на языках Python и C++. Блок с UI вспомогательный и позволяет через web-браузер управлять роботом (как физическим, так и симулятором), проводить эксперименты и управлять модулями, запущенными в ROS.

Сцена с роботом содержит модель робота (описание в URDF), модель робота была взята готовая для UR5, и доработана для решения задач (были добавлены дополнительные узлы, контроллеры и т.п.). Модель робота нуждалась в доработке, потому что не соответствовала физическим параметрам робота, находящегося в лаборатории. Эта модель была откалибрована и поэтому гарантируется, что эксперименты в симуляторе можно легко перенести на реального робота.

На сцену для симулятора добавляется модель робота, а также другие препятствия с разными физическими свойствами. К роботу подключена камера и добавлен контроллер для получения информации о глубине.

Модуль укрупнения препятствий принимает на вход данные с PointCloud и преобразует его в другой формат - параллелепипеды. Для облегчения задачи проведения экспериментов была разработана библиотека препятствий.

Вместо данных с симулятора можно использовать реальные данные с робота, для этого написан отдельный модуль.

Узел ROS содержит:

- модуль загрузки сцены;
- модуль планирования;
- модуль сбора датасета.

Модуль загрузки сцены получает информацию о модели робота и о препятствиях на сцене. Он загружает ее и при необходимости преобразует для дальнейшего планирования на сцене.

Модуль планирования подключает к себе все необходимые узлы: библиотеки Tesseract-Robotics для управления TrajOpt, библиотеки MoveIt для управления RRTConnect и другими классическими методами планирования, модуль для сбора датасета. Он может обращаться к модулю предсказания траектории RNN и модулю расчета кинематики.

Отдельно расположен модуль для обучения модели: он напрямую не связан ни с одним другим модулем, но использует данные, которые сформированы в узле ROS с помощью модуля планирования и модуля загрузки сцены.

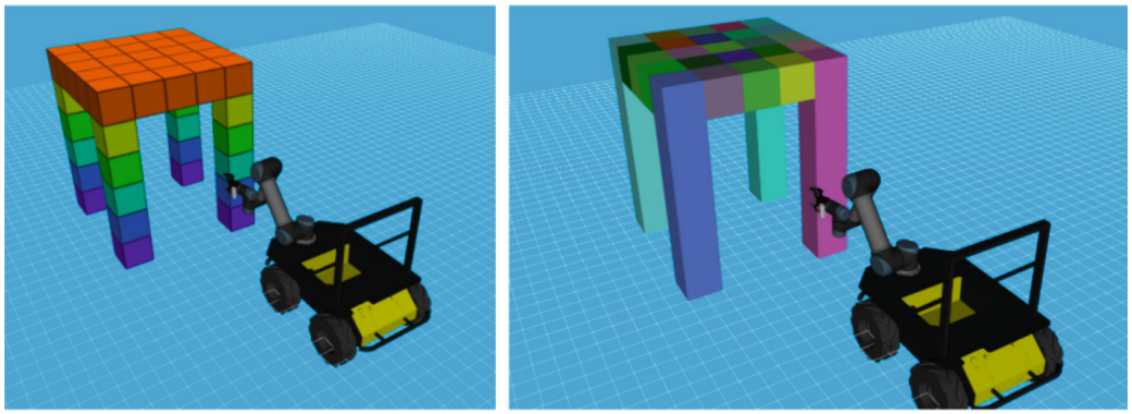


Рис. 14: Octomap (слева) и объединение кубов по оси Z из Octomap (справа)

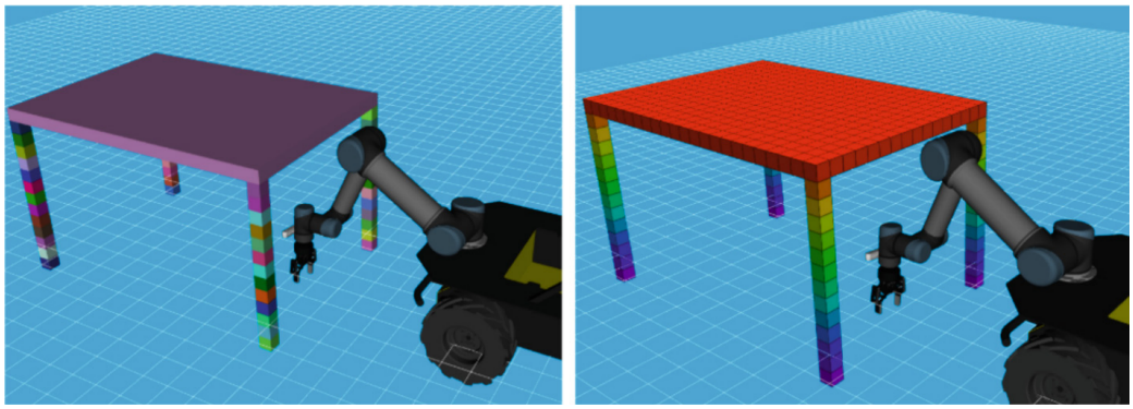


Рис. 15: Octomap (справа) и объединение кубов по осям X, Y (слева)

3.3 Представление препятствий

В рамках предложенного метода ключевым элементом является представление препятствий, так как оно напрямую влияет на эффективность работы алгоритма TrajOpt. Существующие варианты представления, такие как Mesh, Octomap и PointCloud, имеют значительные недостатки, которые могут снизить качество планирования траектории роботизированного манипулятора. Например, снижение точности планирования связано с затратами на приближение препятствий к более удобным для обработки формам, такими как параллелепипеды или эллипсоиды. Повышенная вычислительная сложность, возникающая из-за необходимости обработки мешей, также приводит к увеличению времени планирования.

Пример объединения кубов показан на рисунках 14, 15 и 16.

Недостатки использования существующих представлений препятствий:

- меши обладают высокой точностью благодаря высокой детализации объектов, однако их разбор требует много вычислений;

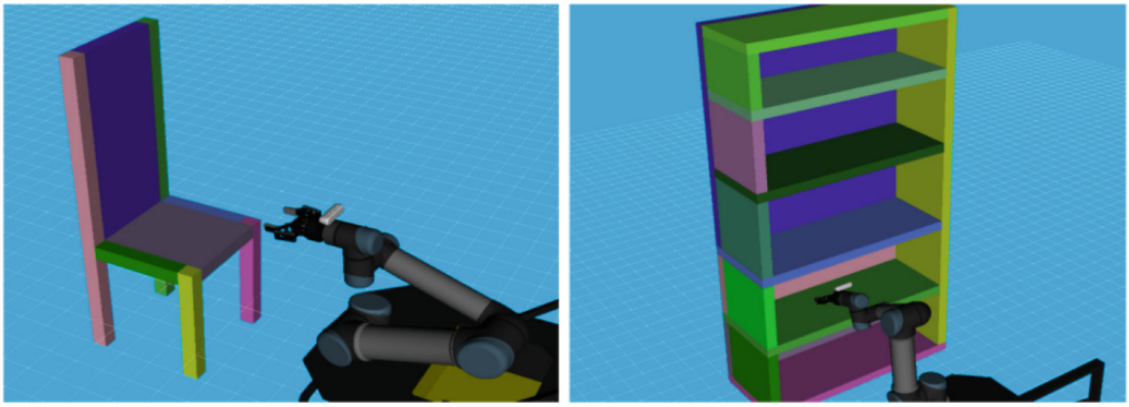


Рис. 16: Объединение кубов по осям X, Y, Z

- аппроксимация препятствий до кубов и эллипсоидов (1 препятствие = 1 куб или 1 эллипсоид) влечет за собой большие потери точности, что может приводить к неправильным результатам при планировании (не нахождение траектории там, где она есть, и наоборот);

- Octomap является наиболее распространенным способом представления препятствий. Указывая разные размеры кубов, он позволяет достичь различной точности приближения, но затрудняет вычисления при планировании траектории.

Предложено улучшить метод представления препятствий в виде Octomap и адаптировать его для задач планирования траектории с манипулятором. Для этого необходимо отфильтровать данные с датчиков, которые находятся за пределами досягаемости манипулятора робота. Информация об препятствиях получена с данных глубинной камеры, установленной на захвате. Камера предоставляет данные о препятствиях в виде облака точек. Точки, расположенные за пределами диапазона манипулятора, фильтруются. Затем облако точек преобразуется в Octomap. Ширина куба Octomap устанавливается вручную и зависит от требуемой точности. Затем необходимо преобразовать PointCloud в Octomap и объединить кубы в параллелепипеды.

Чтобы получить информацию об препятствиях, можно использовать данные симулятора. Пример сцены в симуляторе Gazebo и изображение с установленной на захвате камерой RealSense показаны на рисунке 17. Преобразование данных с камеры симулятора сначала в Octomap, а затем в увеличенные препятствия показано на рисунке 18.



Рис. 17: Сцена из симулятора Gazebo: данные с камеры RealSense (слева) и фрагмент сцены (справа)

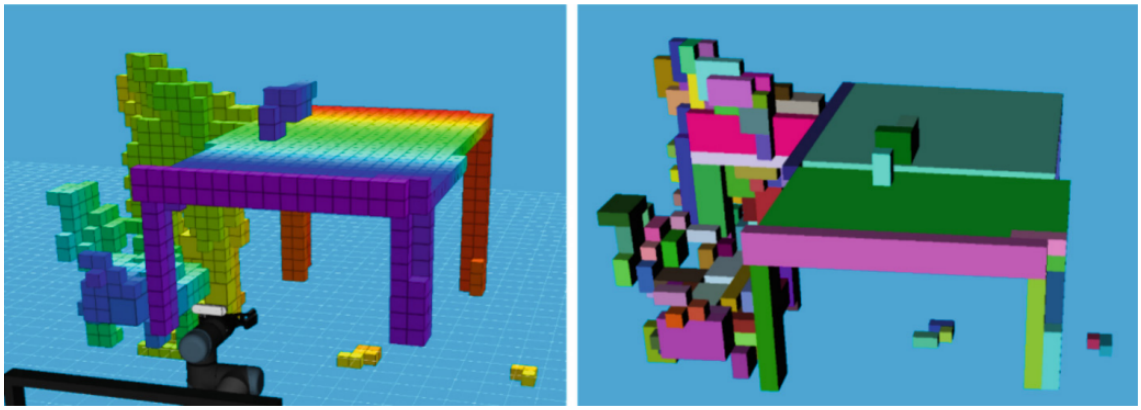


Рис. 18: Octomap для стола (слева) и объединение кубов по X, Y, Z (справа)

Предложено использовать другое представление препятствий вместо Octomap: кубы объединить в параллелепипеды. Для проведения экспериментов была разработана библиотека препятствий, в которую вошли такие препятствия, как: стол, стул, корзина, стена, шкаф (см. рисунок 19).

3.4 Генерация начальной траектории

Важной частью процесса планирования траектории является генерация начальной траектории (начального приближения траектории), которая служит исходной точкой для дальнейшей оптимизации. В данном разделе рассмотрен подход к созданию начальной траектории с учетом текущего положения манипулятора и его целевого положения. Ключевым аспектом будет использование рекуррентной нейронной сети (RNN) с LSTM-ячейками, которая позволяет эффективно обрабатывать последовательные данные, отражающие изменение конфигурации манипулятора.

Тем не менее, несмотря на преимущества RNN, у этого подхода существуют и

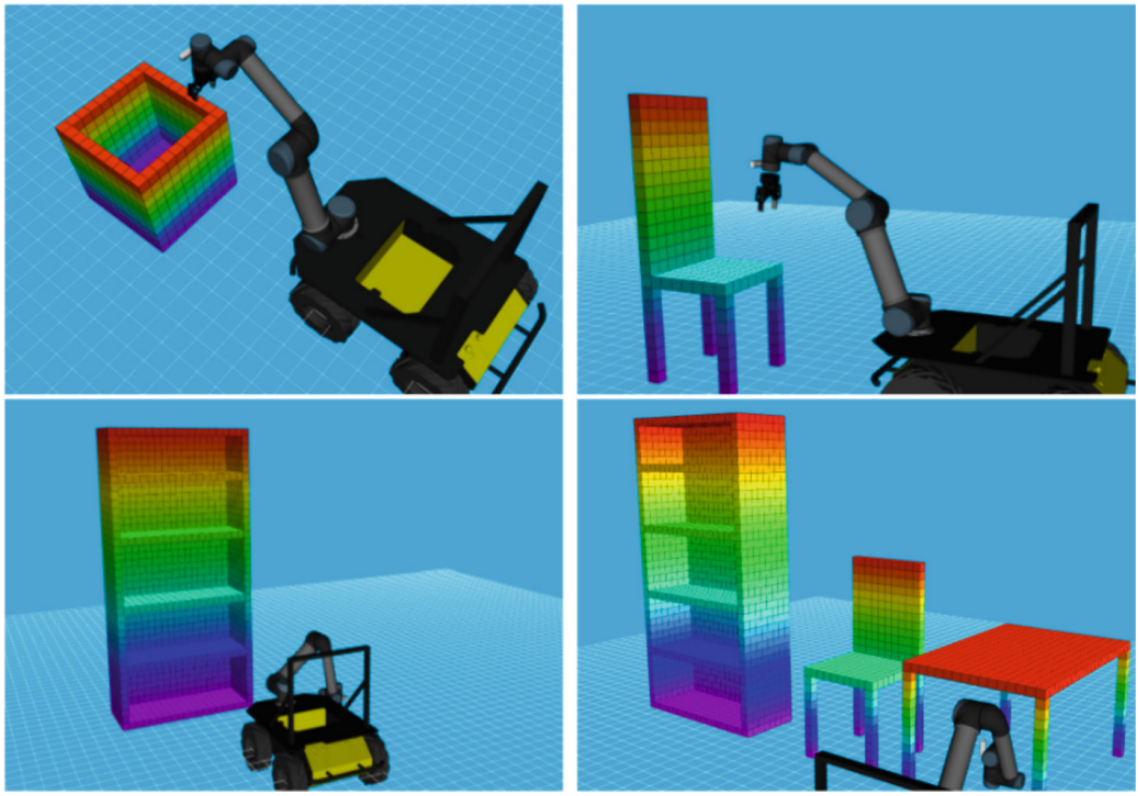


Рис. 19: Библиотека объектов

недостатки, такие как зависимость от исторических данных и сложность в обучении на больших последовательностях, что обеспечивает низкое качество предложенных траекторий. Чтобы преодолеть эти ограничения, будет также внедрен метод обучения с подкреплением, основанный на подходе Actor-Critic. Этот метод позволит улучшить адаптивность системы к изменениям в окружающей среде и повысить качество формирования начальной траектории.

В результате генерируемая траектория будет представлять собой более оптимизированный и информированный путь, что станет основой для дальнейшей оптимизации с использованием алгоритма TrajOpt. Данные для обучения будут включать информацию о сцене, начальном и целевом положениях манипулятора, а также предыдущие успешные траектории, что позволит значительно улучшить процесс планирования.

RNN (Recurrent Neural Network)

В рамках работы рассматривается использование рекуррентной нейронной сети (RNN) с LSTM для генерации начальной траектории манипулятора. Этот нейросетевой подход позволяет эффективно обрабатывать последовательные данные, что способствует созданию более точных и адаптивных траекторий, учитывающих изме-

нения в окружающей среде. В результате обучения RNN будет разработана модель, способная генерировать начальные траектории для последующей оптимизации.

Параметры сети:

- вход - тензор (последовательности произвольной длины с 6 признаками);
- слой маскирования для исключения значений, равных 0;
- слой LSTM с 64 нейронами;
- слой TimeDistributed, который применяет плотный слой (Dense) с 6 выходами ко всем временным шагам выходных данных слоя LSTM;
- используется оптимизатор Adam, функция потерь MSE.

Архитектура модели представлена на рисунке 20.

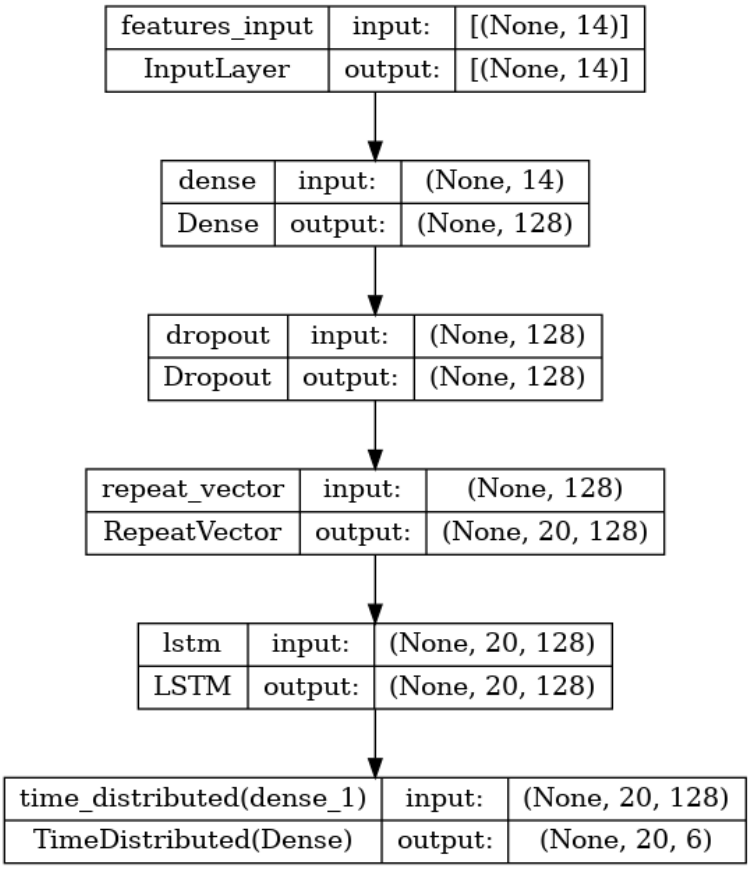


Рис. 20: Модель RNN

Данные для обучения:

- сцена в виде Octomap;

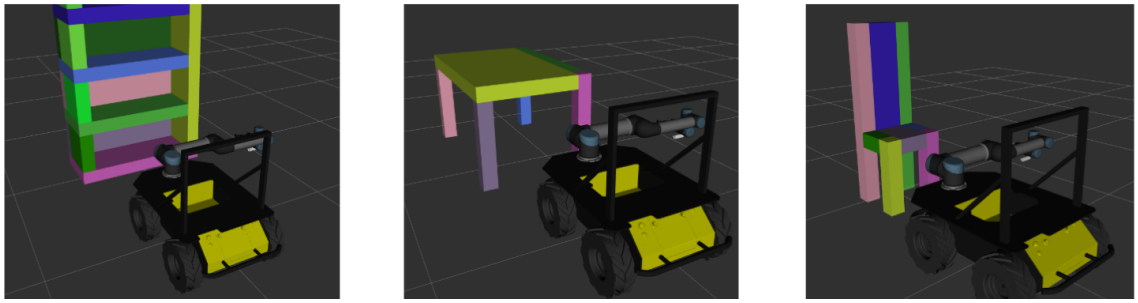


Рис. 21: Сцены для сбора датасета (шкаф, стол, стул)

- начальное и целевое положение робота (6 джоинтов);
- траектория, сгенерированная с помощью RRTConnect или TrajOpt.

Сцены для сбора данных с использованием RNN представлены на рисунке 21. Всего было собрано 3 датасета для обучения:

- Собран вручную (для обучения 218 сцен, для валидации - 81);
- Сгенерирован в зоне досягаемости робота (2663 сцены для обучения и 296 для валидации, много положений расположены далеко от препятствий);
- Сгенерирован в непосредственной близости от препятствий, как показано на рисунке 22 (759 для обучения и 84 для валидации).

Тем не менее, в этом подходе были выявлены несколько проблем. Сгенерированные сетью RNN траектории не отличались разнообразием на первых этапах: она предлагала пути в ограниченных зонах, которые успешно оптимизировались с помощью TrajOpt, обеспечивая 100% эффективность. Однако выяснилось, что такие траектории зачастую проходили далеко от препятствий, что противоречило условиям датасета, где начальные и конечные точки расположены в непосредственной близости от них, что не позволяло провести прямую линию между ними без пересечений (рисунок 23). На практике, несмотря на то что TrajOpt получает начальные и целевые положения робота, он может проигнорировать эти значения. Таким образом, необходимо было учитывать эти данные в предсказанной траектории.

Обновление алгоритма обучения RNN было нацелено на создание более разнообразных траекторий, но это привело к прямо противоположным результатам: уровень успеха упал до 0%. При внимательном анализе стало очевидно, что новые траектории зачастую были невыполнимыми, что подчеркивало необходимость тщательной интеграции требований к препятствиям в процессе генерации.

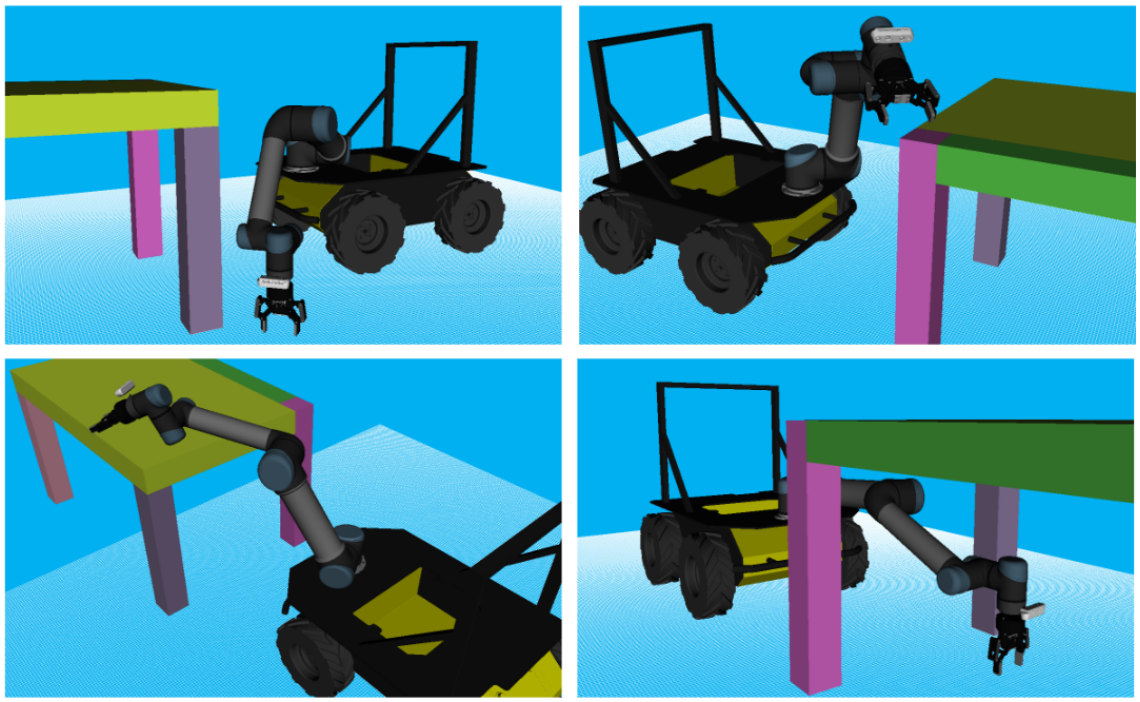


Рис. 22: Определение границ в пространстве для сбора датасета

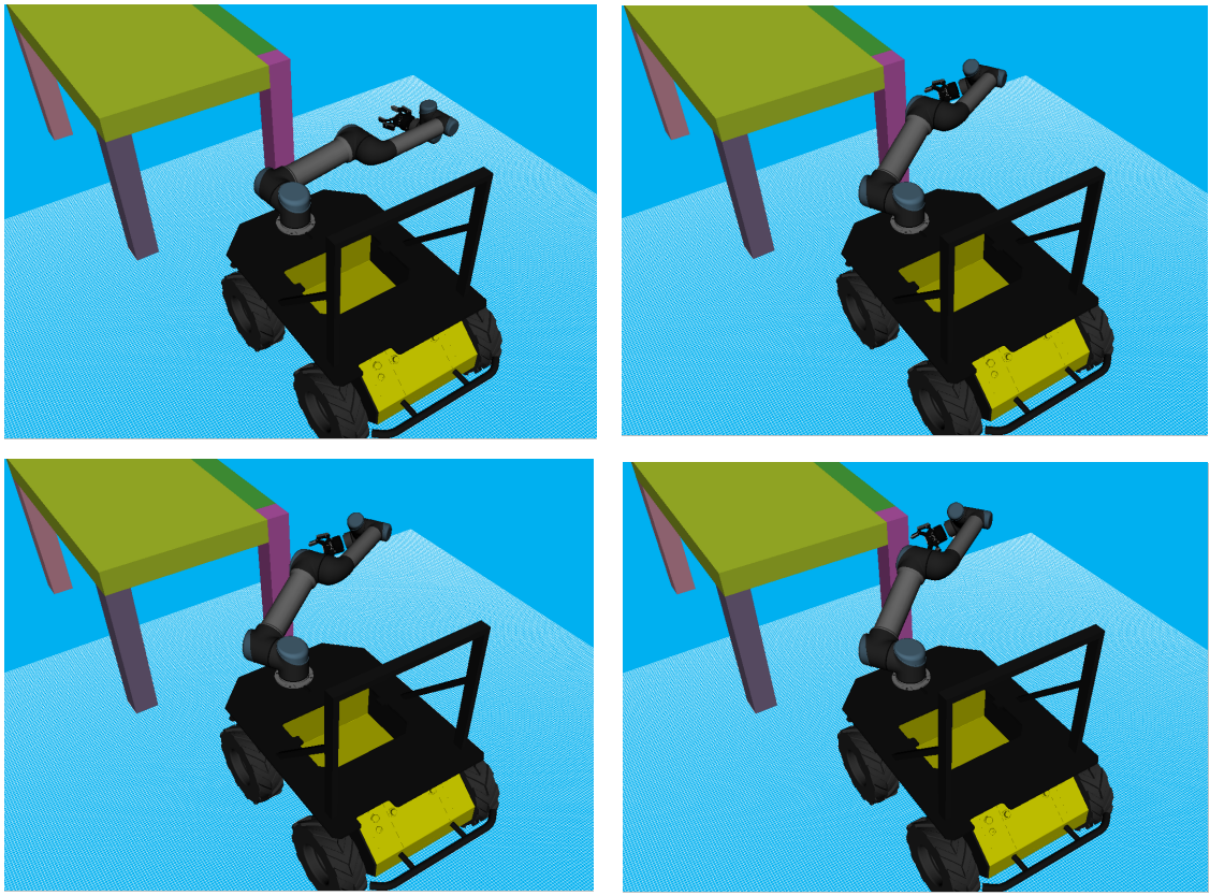


Рис. 23: Робот последовательно принимает положения из траектории, сгенерированной RNN

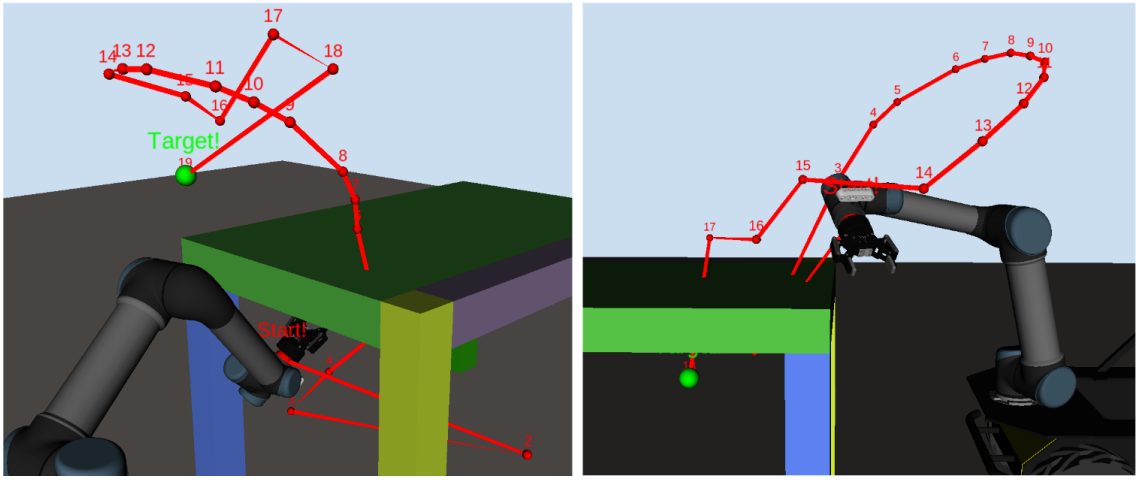


Рис. 24: Траектории, построенные RNN на начальном этапе

Проблемы предсказанных траекторий:

- пересекают препятствия;
- построены хаотично;
- разная длина шага;
- могут перескакивать цель (траектория может быть построена рядом с целью, а потом развернута в противоположную сторону);
- могут предлагать невыполнимые траектории (например, обогнуть препятствие с другой стороны, где робот подъехать не может, как показано на рисунке 29).

Как можно решить эти проблемы:

- исключить невыполнимые траектории, обеспечить выход робота из-под препятствия (на этом шаге кажется наиболее рациональным, чтобы в список траекторий добавилось положение, в котором робот расположен в сложенном состоянии над подвижной платформой, как показано на рисунке 25);
- упорядочить точки внутри траектории в направлении уменьшения расстояния до цели;
- из упорядоченного списка исключить пары, отрезок между которыми пересекает препятствия;
- удалить дублирующиеся точки (и пары точек).

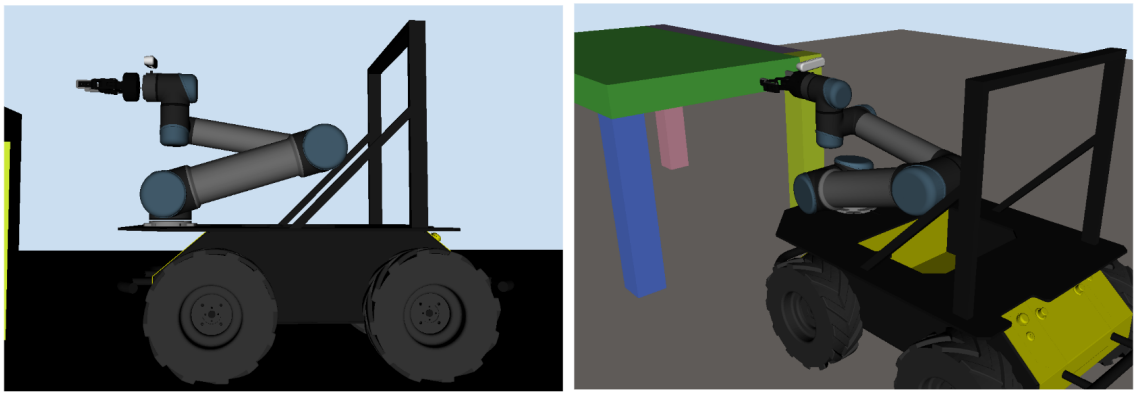


Рис. 25: Положение робота над платформой

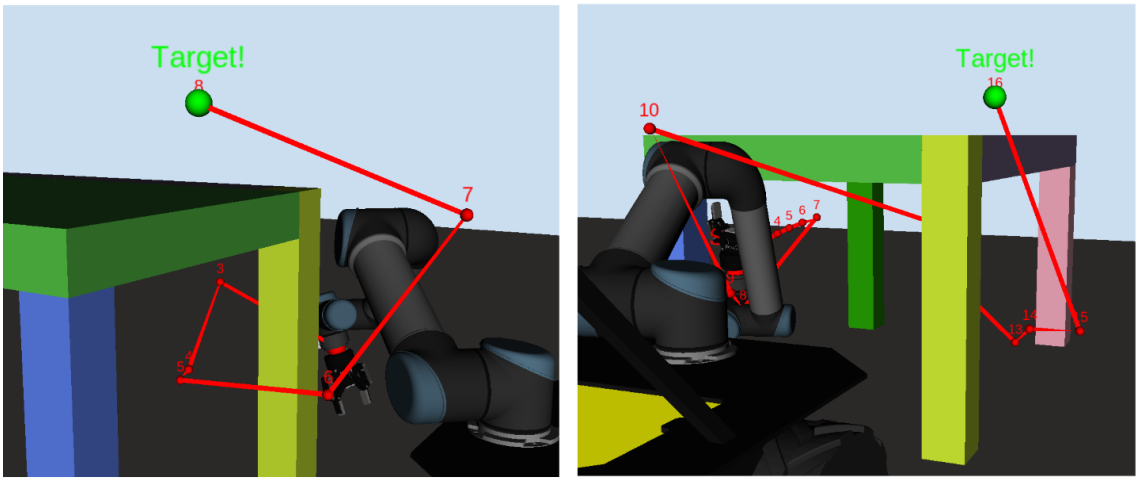


Рис. 26: Траектории, построенные с помощью RNN и улучшенные, но не выполнимые роботом

После внедрения этих доработок по-прежнему некоторые траектории генерировались таким образом, что из невозможно было выполнить. Как, например, на рисунке 26.

Предпринятые меры дали результат предсказания на валидационом датасете 24%. Однако по-прежнему некоторые траектории генерировались таким образом, что их невозможно было выполнить. Как, например, на рисунке 26. Некоторые траектории выглядели выполнимыми, но TrajOpt не смог их оптимизировать (рисунок 27). Но были и те, с которыми TrajOpt справился (28).

После внедрения алгоритма для обработки предсказанных траекторий удалось достичь результата выше 0%, однако уровень успеха в 24% оставался крайне низким (у базового решения с RRTConnect - 65%). Несмотря на усилия по улучшению генерируемых путей, RNN продолжала предлагать невыполнимые траектории. Таким образом, хотя данный подход не подходит для использования в качестве предсказателя начальных траекторий, он может служить базовым решением, с которым

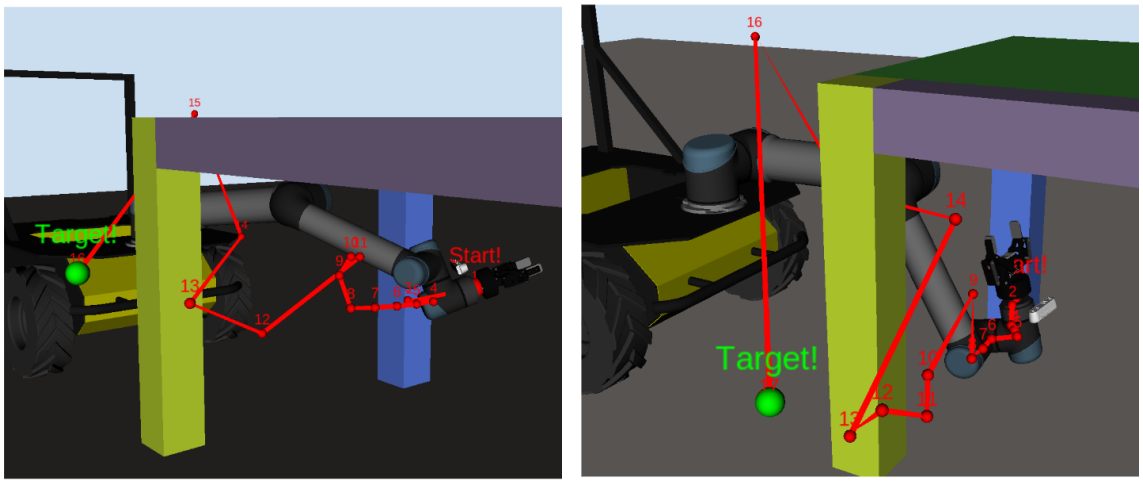


Рис. 27: Траектория, построенная с помощью RNN и улучшенная, не пересекающая препятствий

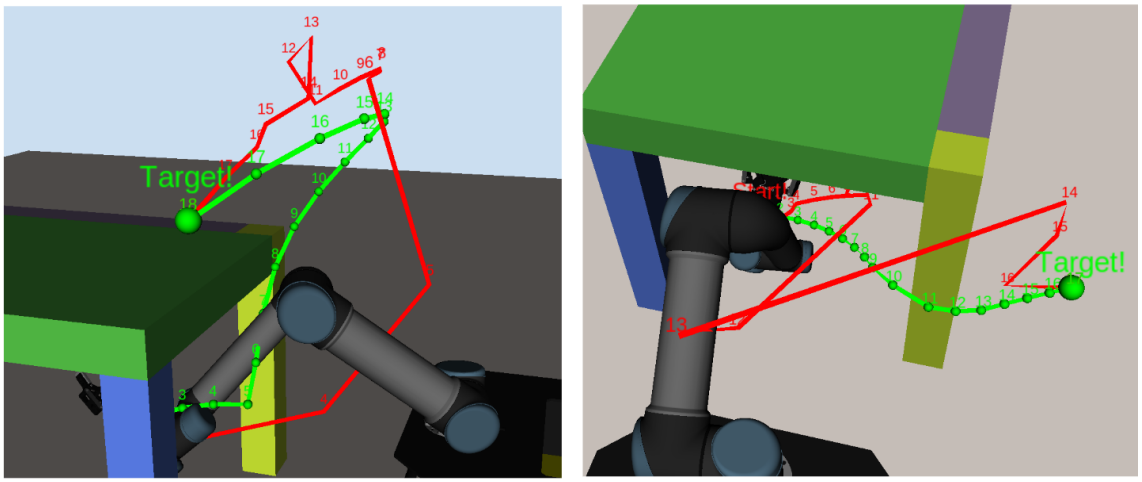


Рис. 28: Траектория, построенная с помощью RNN (красным) и оптимизированная TrajOpt (зеленым)

можно будет сравнить другие подходы. В дальнейшем данные, собранные в процессе формирования датасета для обучения RNN, были успешно применены для оценки эффективности других алгоритмов.

Работа с RNN подсветила ряд проблем, которые есть при планировании траектории:

- траекторию необходимо строить пошагово, начиная с точки старта;
- необходимо учитывать, что робот находится на неподвижной базе, и нельзя оперировать только понятием точек в декартовом пространстве;
- понятием джоинтов тоже сложно оперировать, поскольку недостаточно предсказать джоинты, необходимо, чтобы существовали решения обратной кинема-

тики;

- при планировании в среде с препятствиями (например, рядом с ножкой стола), робот может ее обогнуть с двух сторон, но чаще всего один из вариантов приведет к невыполнимой траектории (как на рисунке 26).

Для построения траекторий подходит метод обучения с подкреплением (Reinforcement Learning, RL), который позволяет более эффективно справиться с упомянутыми выше проблемами.

RL (Reinforcement Learning)

Итак, вернемся к цели. Нужно найти траекторию, удовлетворяющую ограничениям, которые обеспечивают безопасное перемещение робототехнического манипулятора от начальной позиции к целевой с учетом окружающей среды. При планировании траектории важно учесть:

- начинать строить траекторию стоит с точки старта, постепенно отдаляясь от нее и приближаясь к цели;
- важно, чтобы траектория проходила в обход препятствий;
- важно, чтобы траектория была выполнимой;
- чем меньше длина траектории, тем лучше.

Для успешного планирования траектории робототехнического манипулятора необходимо учитывать не только возможность достижения каждой цели по отдельности, но и общую реальность выполнения всей траектории в заданной среде. Сложность заключается в том, что манипулятор, перемещающийся в пространстве, должен избегать столкновений как со стационарными препятствиями, так и самими собой. В этом контексте важно сформулировать понятие выполнимости траектории.

Выполнимость траектории подразумевает, что робот может последовательно достигать каждой точки на своей траектории, при этом исключая возможность самоколлизий и коллизий с окружающими объектами. Учитывая, что манипулятор располагается на подвижной платформе и предполагается его движение вперед к препятствиям, необходимо определить безопасную зону. Эта зона позволяет избежать опасных ситуаций, обеспечивая, чтобы манипулятор не пересекался с другими объектами и не подвергал себя риску.

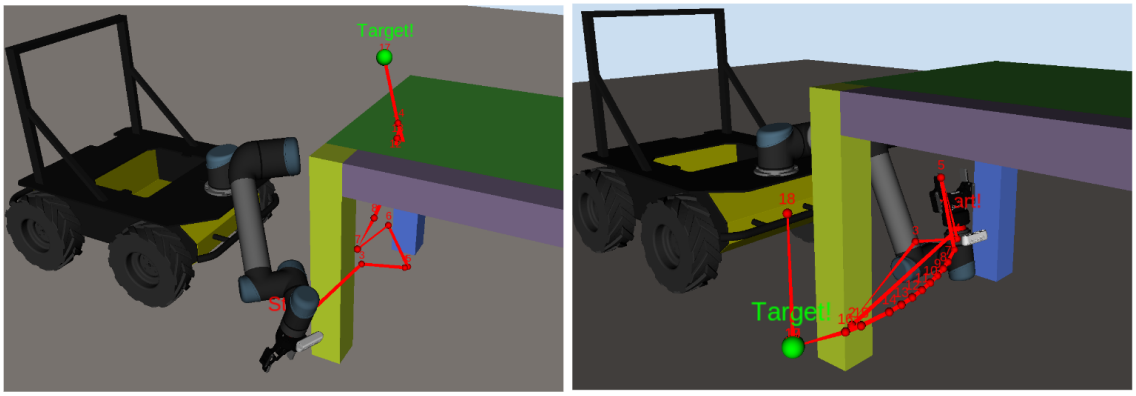


Рис. 29: Траектория проходит под столом

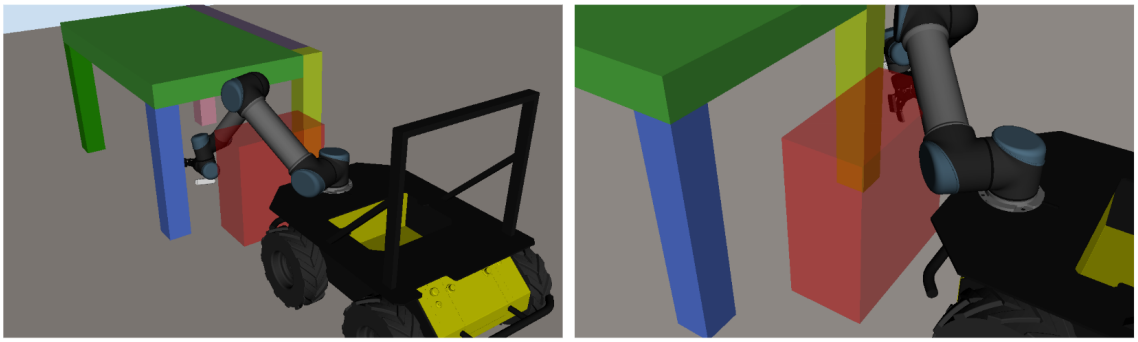


Рис. 30: Обязательная для планирования зона (красная полупрозрачная)

Важность введения такой зоны иллюстрируется на примере рисунка 29, где можно увидеть, что, хотя технически условия избегания коллизий соблюдаются, робот не может полностью пройти заданную траекторию. Для успешного выполнения задачи требуется, чтобы манипулятор смог располагаться так, чтобы он всегда мог приходить в безопасные позиции относительно своей платформы. Таким образом, возникает необходимость в обязательной зоне, через которую должна проходить предсказанная траектория, что будет представлено на следующем рисунке 30.

На основании вышеизложенного формулируется теорема о выполнимости траектории, которая связывает понятие безопасной зоны с условием выполнения задачи манипулятором.

Теорема 3.1. *Пусть* $Traj(s_t)$ *— это траектория манипулятора с 6 степенями свободы, которая начинается в* s_{start} *и заканчивается в* s_{target}*. Если для любого положения* $s \in Traj(s_t)$ *выполняется условие* $Z_b \subset Traj(s_t)$*, то траектория* $Traj(s_t)$ *является выполнимой и безопасной для манипулятора, даже учитывая возможность скрытия некоторых его частей за препятствиями.*

Доказательство. Для доказательства утверждения о выполнимости и безопас-

ности траектории $\text{Traj}(s_t)$, будем использовать математическую индукцию по количеству шагов k траектории, где k представляет количество состояний на пути от s_{start} до s_{target}.

База индукции: для $k = 1$ (т.е. только начальное состояние s_{start}). Если $s_{start} \in Z_b$, то, по определению, $C(s_{start}) = 0$, что означает безопасность начального положения. Таким образом, база индукции выполняется.

Шаг индукции. Предположим, что для некоторого $k = n$ верно, что траектория $\text{Traj}^{(n)}(s_t)$ является безопасной, т.е. для всех состояний s_i в траектории, $Z_b \subset \text{Traj}^{(n)}(s_t)$ приводит к тому, что $C(s_i) = 0$. Теперь рассмотрим ситуацию, когда мы добавляем еще одно состояние s_{n+1} к нашей траектории:

$$\text{Traj}^{(n+1)}(s_t) = (s_{start}, s_1, s_2, \ldots, s_n, s_{n+1}).$$

При предположении, что s_{n+1} подразумевает переход от s_n, и что $Z_b \cap \text{Traj}(s_t) \neq \emptyset$, нам нужно доказать, что и это состояние s_{n+1} безопасно.

Поскольку s_n было безопасным, существует достаточное пространство в Z_b, которое обеспечит переход к s_{n+1}. Если s_{n+1} также попадает в область Z_b, то по аналогии, $C(s_{n+1}) = 0$.

Тем самым, если можно безопасно перейти от состояния s_n к состоянию s_{n+1} и всегда оставаться в области Z_b, это подтверждает, что:

$$C(s_{n+1}) = 0.$$

.

По принципу математической индукции, утверждение о том, что если траектория $\text{Traj}(s_t)$ проходит через обязательную зону безопасности Z_b, то она не имеет коллизий для всех состояний от s_{start} до s_{target}, верно для всех k. Таким образом, заключаем, что траектория является выполнимой и безопасной.

Выполнимость и безопасность траектории с учетом обязательной зоны безопасности являются ключевыми аспектами оптимизации движения роботизированного манипулятора. Этого достичь можно благодаря применению подхода, который эффективно настраивает стратегии действий в сложных и динамичных средах, что и будет основой следующего этапа. Одним из таких методов как раз и является алгоритм обучения с подкреплением на основе Actor-Critic, который позволяет оптимизировать действия в контексте высказанных ограничений.

Этот алгоритм подходит для решения задачи по следующим причинам:

- Actor (агент) отвечает за выбор действия на основе текущего состояния, что позволяет ему обучаться эффективно путем оптимизации стратегий. A Critic оценивает выбранные действия, предоставляя обратную связь об их качестве, что позволяет устранить недостатки выбора действий, предлагая целенаправленное обучение.

- В отличие от чисто policy-based методов, таких как REINFORCE, где обучение может быть нестабильным, Actor-Critic объединяет преимущества value-based и policy-based подходов, что обеспечивает более стабильное и надежное обучение.

- Алгоритм Actor-Critic эффективно работает в задачах с непрерывным пространством состояний и действий, что критически важно для робототехнических приложений, где управление манипуляторами обычно осуществляется дома в непрерывных пространствах.

- Обратная связь от критика позволяет актеру лучше понимать, какие действия приводят к хорошим результатам, что повышает качество разработки стратегии. Критик предоставляет значения состояния (или действия), которые уменьшают вариативность градиента и улучшают общую эффективность обучения.

- Actor-Critic позволяет агенту совершенствовать свои действия на основе полученного опыта, что делает его подходящим для решения задач, где требования могут изменяться или когда не все данные заранее известны, что характерно для многих задач управления агентов.

UML диаграмма последовательности подхода Actor-Critic изображена на рисунке 31.

Алгоритм Actor-Critic поддерживает два семейства параметров:

- Critic: параметры w, относящиеся к Q-функции. Эти параметры обновляются согласно методу временных различий TD(0), что позволяет оценивать полезность состояний и действий в процессе обучения.

- Actor: параметры θ, которые представляют стратегию. В отличие от Critic, они обновляются с учетом оценок, предоставляемых Critic. Это происходит через градиентный метод, известный как policy gradient.

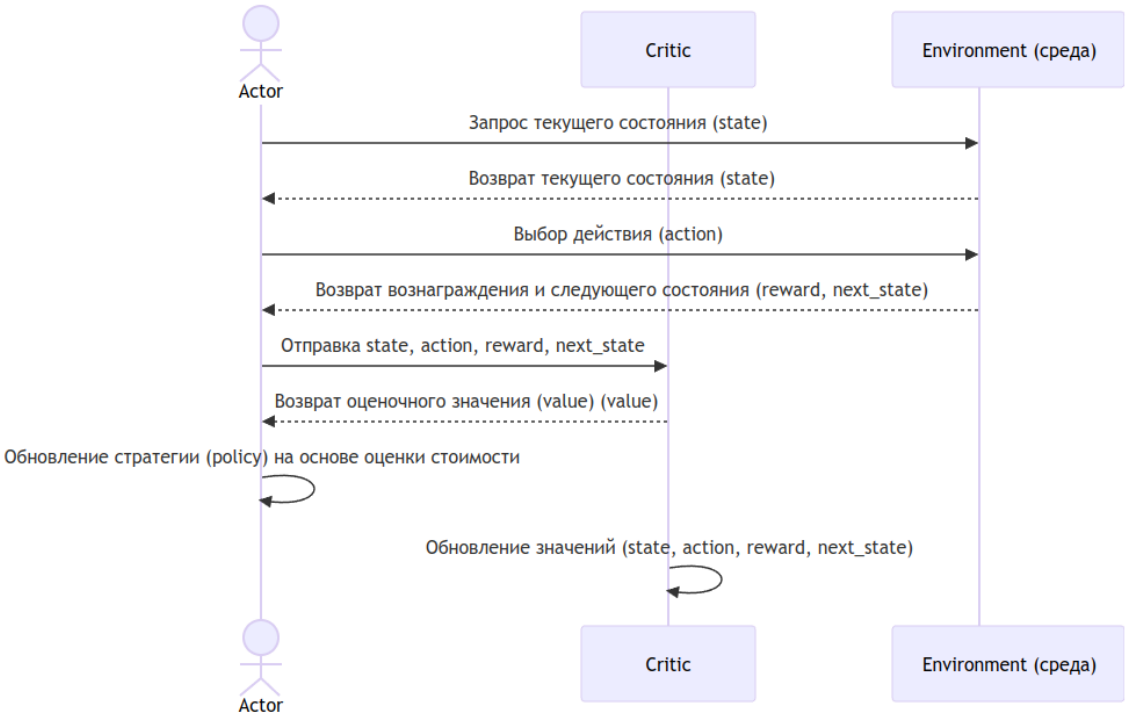


Рис. 31: Диаграмма последовательности Actor-Critic

В рамках данного подхода важно учитывать функцию преимущества (Advantage function), которая описывает, насколько выбранное действие a в состоянии s лучше или хуже, чем ожидалось, основываясь на Q-функции и функции стоимости состояний:

$$A(s, a) = Q(s, a) - V(s)$$

Где $Q(s, a)$ определяется как сумма мгновенной награды r и дисконтированного значения функции стоимости следующего состояния $V(s')$:

$$Q(s, a) = r + \gamma V(s')$$

Следовательно, функция преимущества может быть выражена следующим образом:

$$A(s, a) = r + \gamma V(s') - V(s)$$

Теперь, чтобы улучшить стратегию Actor, мы можем использовать градиент, который дает следующее приближение для шага обновления:

$$\nabla J_{actor} \approx \frac{1}{N} \sum_{i=1}^{N} \sum_{s,a \in z_i} \nabla \log \pi_\theta(a|s) A(s)$$

Где $\pi_\theta(a|s)$ - это стратегия, которая определяет вероятность выбора действия a в состоянии s. Таким образом, мы можем обновить параметры Actor, используя информацию о преимуществах, чтобы сделать выбор действий более целенаправленным.

Это позволяет агенту динамически адаптировать свои действия на основе результатов, что делает алгоритм Actor-Critic подходящим для сложных и динамичных сред, таких как управление робототехническими манипуляторами.

С другой стороны, для обновления Critic мы используем функцию потерь, которая определяется как:

$$L_{critic} \approx \frac{1}{N} \sum_{i=1}^{N} \sum_{s,a \in z_i} (V_\theta(s) - [r + \gamma V(s')])$$

Здесь $[r+\gamma V(s')]$ представляет собой математическое ожидание для $V_\theta(s)$. Таким образом, мы обучаемся предсказывать $V_\theta(s)$ таким образом, чтобы минимизировать разницу между предсказанием стоимости и фактическим ожидаемым значением.

Поскольку цель определена как найти траекторию, удовлетворяющую ограничениям, которые обеспечивают безопасное перемещение робототехнического манипулятора от начальной позиции к целевой с учетом окружающей среды. Ее можно сформулировать так:

$$\max_A \sum_{t=0}^{T} \gamma^t R(s_t, a_t)$$

где $R(s_t, a_t)$ — функция награды, T — общее количество шагов, γ — коэффициент дисконтирования.

Для решения задачи введем обозначения (параметры).

- Состояния:
  - $s_t \in \mathbb{R}^n$ — вектор состояний, описывающий текущее положение манипулятора.
  - s_{start} — начальное состояние манипулятора.
  - s_{target} — целевое состояние манипулятора.

- Действия: $a_t \in \mathbb{R}^m$ — вектор действий, представляющий перемещения манипулятора в трехмерном пространстве (x, y, z).

- Награда: $R(s_t, a_t)$ — функция награды, которая оценивает, насколько безопасным и эффективным было действие на текущем шаге.

- Ограничения: $C(s_t)$ — условия, которые должны быть выполнены для того, чтобы состояние s_t считалось допустимым (например, отсутствие коллизий с препятствиями).

При переходе между состояниями необходимо проверять, не пересекаются ли перемещения манипулятора с препятствиями:

$$C(s_t) = \begin{cases} 0, & \text{если } s_t \text{ безопасно} \\ -\infty, & \text{если } s_t \text{ имеет коллизию} \end{cases}$$

Также важно, чтобы сначала путь строился до обязательной зоны безопасности, а только потом до цели. Условие может быть записано как:

$$\text{Если } Z_b \subset \text{Traj}(s_t), \text{ тогда траектория считается правильной,}$$

где Z_b — это обязательная зона безопасности, а $\text{Traj}(s_t)$ — траектория, описывающая движение манипулятора в пространстве состояний.

Формально описать задачу можно следующим образом:

$$\max_{\pi} \mathbb{E}_{X \sim p(X)} \left[\mathbb{E}_{(s_i, a_i) \sim p_\pi} \left[\sum_{i=0}^{N-1} \gamma^i \cdot R_i(s_i, a_i) \mid Z_b \subset \text{Traj}(s_t) \right] \right]$$

Где:

- π — стратегия или policy (выбор действий), которую обучаем.

- $\mathbb{E}_{X \sim \rho(X)}$ — математическое ожидание, взятое по распределению $\rho(X)$ (например, распределение состояний среды, которое может быть известным или изученным).

- $\mathbb{E}_{(s_i, a_i) \sim \rho_\pi}$ — математическое ожидание, взятое по всем парам состояний и действий, соответствующим выбранной стратегии π.

- s_i — состояние агента на шаге i.

- a_i — действие, выбранное агентом на шаге i, согласно стратегии π.
- $R_i(s_i, a_i)$ — функция награды, которая оценивает, насколько хорошим было действие a_i в состоянии s_i.
- γ — коэффициент дисконтирования, который определяет, насколько ценятся будущие награды (значение находится в пределах [0, 1]).
- N — количество временных шагов, на которое мы рассчитываем при планировании действий.
- Z_b — это обязательная зона безопасности.
- $\mathrm{Traj}(s_t)$ — траектория движения манипулятора.

Таким образом, решается задача максимизации ожидаемой суммарной награды (вознаграждения) за всё время N при условии, что агент действует согласно стратегии π.

Определение пространства действий: пространство действий определено как разница конфигурации ($a_i = \triangle q_i$) относительно текущей конфигурации (q_i), где $a_{i+1} = q_i + \triangle q_i$ при детерминированной функции перехода T.

Представление пространства состояний:

- пространство состояний (s_i) на определенном временном шаге (s_i) представлено как кортеж, $s_i = (q_i, p_i^{link}, p_i^{target}, z_{env})$;
- p_i^{link} - список поз, представляющих декартовы координаты (6 измерений: комбинация текущего положения и целевого);
- p_i^{target} - целевая поза в декартовом пространстве;
- z_{env} - вектор контекста сцены (с учетом препятствий в виде параллелепипедов).

Функционал цели для задачи следования за путём, требующим минимизации:

$$U[\xi] = U_{\text{pose}}[\xi] + \lambda_1 \cdot U_{\text{obs}}[\xi] + \lambda_2 \cdot U_{\text{smooth}}[\xi],$$

где:

- $U_{\text{pose}}[\xi]$ - ошибка позиции end-effector,

- $U_{\text{obs}}[\xi]$ - стоимость за избегание препятствий,
- $U_{\text{smooth}}[\xi]$ - стоимость за гладкость траектории,
- λ_1 и λ_2 - весовые коэффициенты для балансировки между подцелями.

Оптимизация начинается с генерации начальной траектории ξ_{init}, которая затем уточняется для минимизации $U[\xi]$:

$$\arg\min_{\xi_{\text{init}}} U[\xi_{\text{init}}].$$

Траектория должна удовлетворять ограничениям: пределы суставов, избегание сингулярностей, предотвращение столкновений.

Общая модель вознаграждения на каждом временном шаге i представлена следующей формулой:

$$R_i = R_{\text{task},i} + R_{\text{cstr},i},$$

Модель вознаграждения формулируется для каждого временного шага i и включает в себя:

- вознаграждение за выполнение задачи (R_{task}, i) учитывает ошибки положения и ориентации end-effector по сравнению с целевым путём.
- вознаграждение за соблюдение ограничений (R_{cstr}, i) направлено на минимизацию нарушений кинематических и динамических ограничений.

Предлагаемое решение представляет собой мощный метод обучения с подкреплением, используя подход Actor-Critic для планирования траекторий робототехнического манипулятора. В системе используются две нейронные сети: Actor, который отвечает за выбор действий на основе состояния, и Critic, который оценивает эффективность этих действий. Эта структура позволяет эффективно обучать агента в сложной среде, обеспечивая быстрое и точное принятие решений на основе текущих оценок состояния.

В предложенной архитектуре нейронной сети реализованы две основные компоненты - Actor и Critic, каждая из которых имеет свои особенности и слои.

Класс Actor отвечает за определение стратегии или политики, которую агент будет использовать для выбора действий на основе текущего состояния среды. Архитектура Actor состоит из нескольких полносвязных слоев:

- *Входной слой.* Первый полносвязный слой принимает в качестве входных данных вектор состояния размером 6 (координаты текущего состояния и цели). Это может включать в себя текущие координаты робота и дополнительные параметры, необходимые для принятия решений.

- *Скрытые слои.* первый скрытый слой состоит из 256 нейронов с активацией, заданной функцией ReLU, что помогает в обработке нелинейностей в данных. После первого скрытого слоя используется Dropout с вероятностью 0.2 для регуляризации, что помогает предотвратить переобучение. Второй скрытый слой также имеет 128 нейронов и повторяет структуру, с добавлением активации ReLU и использования Dropout.

- *Выходной слой.* Последний слой формируется в соответствии с количеством доступных действий, содержащихся в action (массив действий). Этот слой не имеет активации, потому что мы применяем softmax в процессе выбора действий.

Класс Critic оценивает действия агента, предоставляя оценку (предполагаемую «ценность») данного состояния или действия. Его архитектура схожа с архитектурой Actor, но имеет определенные отличия:

- Входной слой. Как и в Actor, первый полносвязный слой принимает вектор состояния размером 6.

- Скрытые слои. Первый скрытый слой состоит из 256 нейронов, использует ReLU и Dropout, аналогично Actor. Второй скрытый слой также имеет 128 нейронов и повторяет ту же структуру.

- Выходной слой. Выходной слой состоит из одного нейрона, который возвращает оценку для текущего состояния.

Архитектура нейронной сети описана на рисунке 32.

Класс ActorCriticAgent объединяет и управляет этими двумя сетями. Используются оптимизаторы Adam для обновления весов обоих сетей с заданной скоростью обучения. Агент обрабатывает потери как для Actor, так и для Critic, используя соответствующие функции потерь – SmoothL1Loss для Critic и производную от логарифма для Actor. Введение Target Network позволяет более стабильное обновление

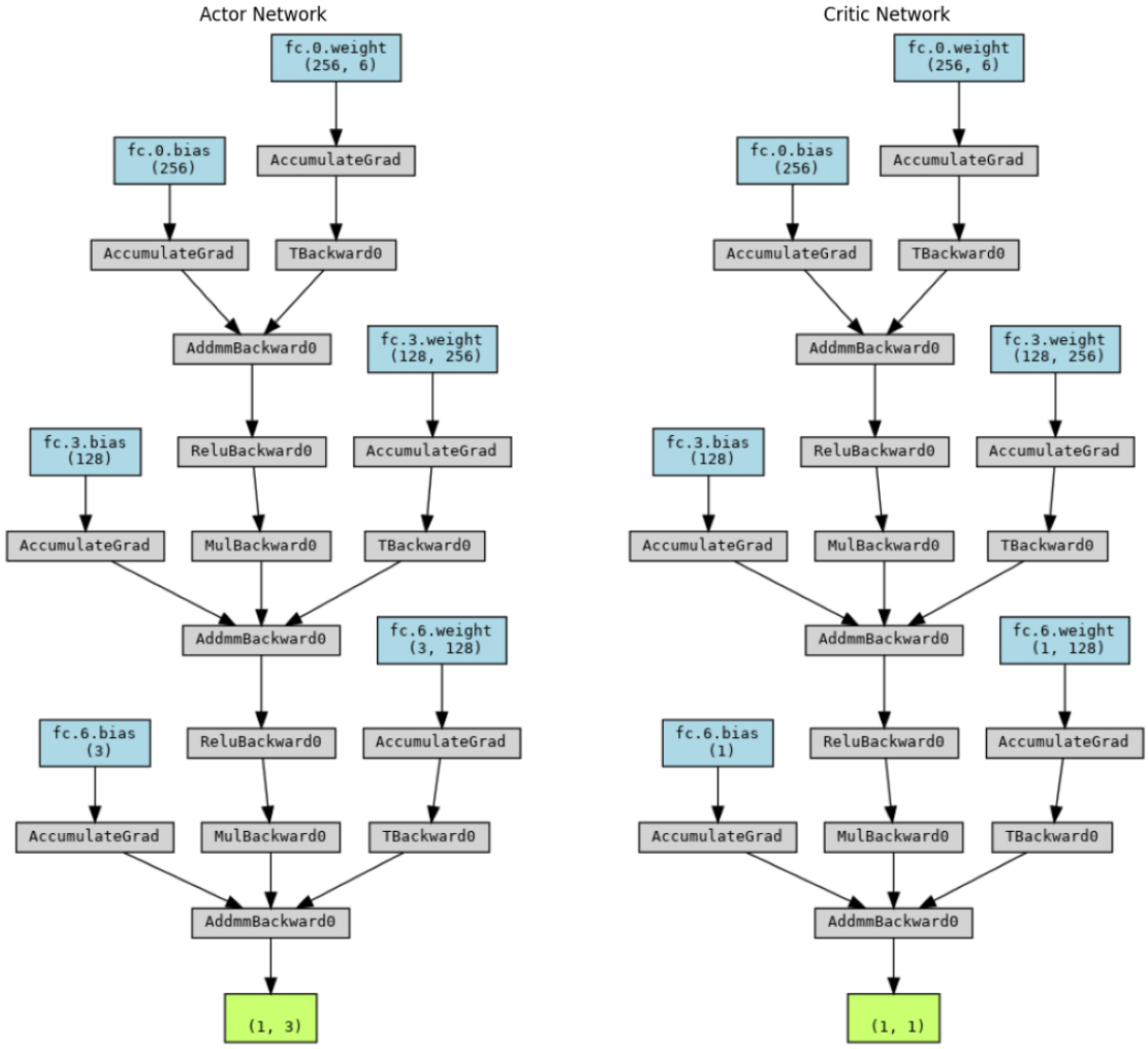


Рис. 32: Архитектура нейронных сетей Actor и Critic

для Critic, уменьшая колебания в процессе обучения. Агент имеет методы для выбора действий на основе текущего состояния и оценки потенциальных последствий каждого действия, включая поддержку случайного выбора действий в процессе обучения (epsilon-greedy). Подход описан в псевдокоде 3.1.

На рисунке 33 изображена схема работы механизма предсказания.

Базовые параметры для обучения. Обучение проводилось в течение 1000 эпизодов, каждый из которых состоял из 500 шагов. На начальном этапе агент в основном сосредоточился на исследовании окружающей среды, выбирая случайные действия из-за повышенного значения параметра ϵ. По мере прогресса в обучении стратегия агента становилась более определенной, и значение ϵ постепенно снижалось. Коэффициент дисконтирования (γ) был установлен на уровне 0.99, что свиде-

Алгоритм 3.1 Обучение Actor-Critic агента

Вход: $episodes, max_steps, actions, required_zone$;

Выход: Обученная модель агента A;

для $episode = 1$ **to** $episodes$
  инициализация: $start_point, target_point$
  $state \leftarrow start_point$
  $total_reward \leftarrow 0$, $path \leftarrow [start_point]$
  для $i = 1, \ldots, max_steps$
    $direction_to_target \leftarrow calculate_direction_to_target(target_point, state)$
    $state_with_direction \leftarrow [state, direction_to_target]$
    $action \leftarrow A.select_action(state_with_direction)$
    $next_state \leftarrow$ calculate_new_state$(state, action)$
    если достигнута цель **то**
      $done \leftarrow true$;
      EXIT;
    $reward \leftarrow$ get_reward$(state, next_state, target_point)$
    $path.append(next_state)$
    $A.update(state_with_direction, action, reward, next_state, done)$
    $state \leftarrow next_state$
    $total_reward \leftarrow total_reward + reward$
  $A.adjust_epsilon()$

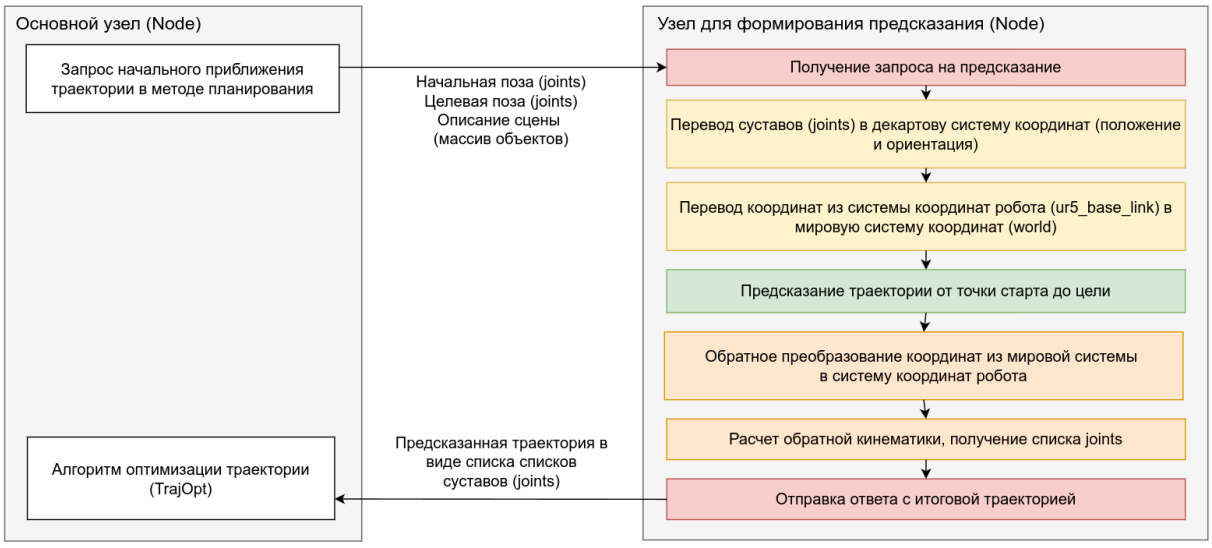


Рис. 33: Схема работы алгоритма предсказания

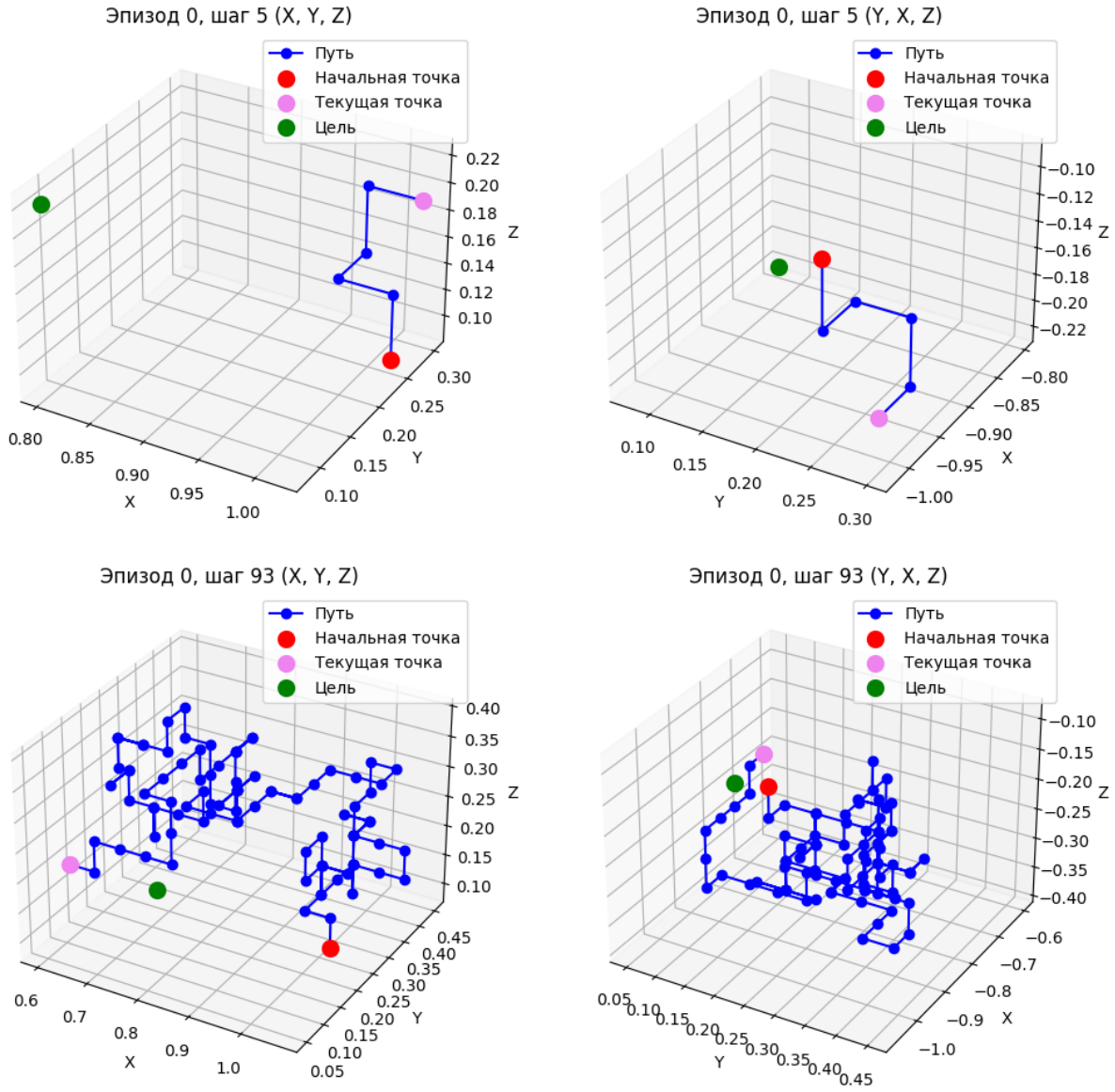


Рис. 34: Построение траектории на этапе обучения (начало обучения)

тельствует о важности горизонта планирования для агента.

Пример траектории (в динамике), построенной во время исследования среды, представлен на рисунке 34. Пример построения длинной траектории показан на рисунке 35. В итоге агент нашел путь, но ему пришлось для этого предпринять много шагов. Это с учетом того, что действия в основном в этом примере выбирались стохастически. Чем больше эпизодов проходило, тем меньше предпочтение отдавалось случайным действиям. Это нашло отражение в траектории. Она становится более предсказуемой и направлена в сторону сначала обязательной безопасной зоны, а затем в сторону цели.

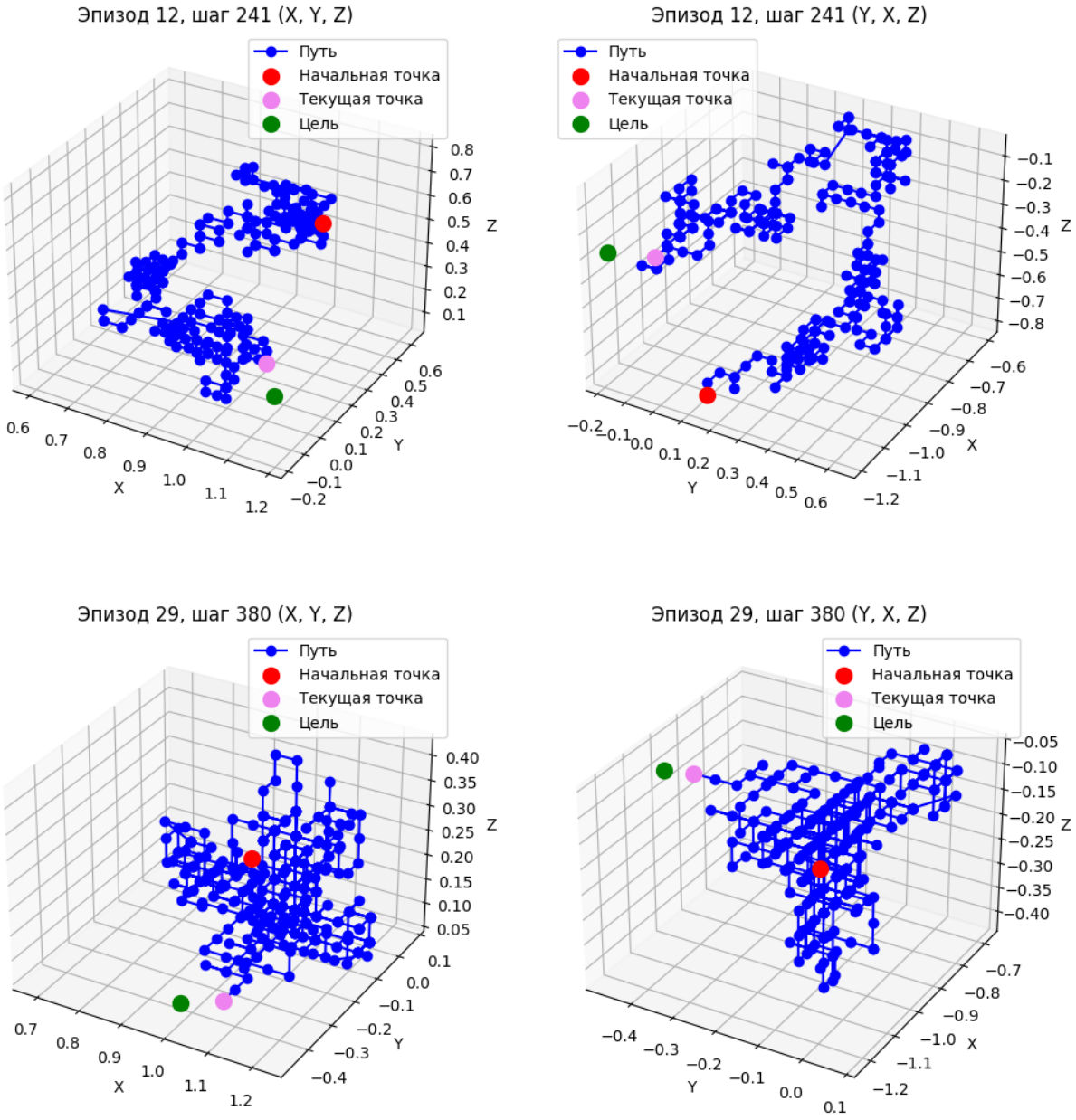


Рис. 35: Построение траектории на этапе обучения (длинные траектории)

4 Вычислительные эксперименты

Для подтверждения выдвинутых предположений были проведены вычислительные эксперименты, целью которых стало тестирование и верификация предложенного метода планирования траекторий манипулятора в условиях взаимодействия с препятствиями. Важно отметить, что данные эксперименты позволили не только визуализировать процесс планирования, но и оценить возможность успешного выполнения заданных траекторий с учетом различных условий окружающей среды.

Эксперименты проводились в среде Robot Operating System (ROS), что обеспе-

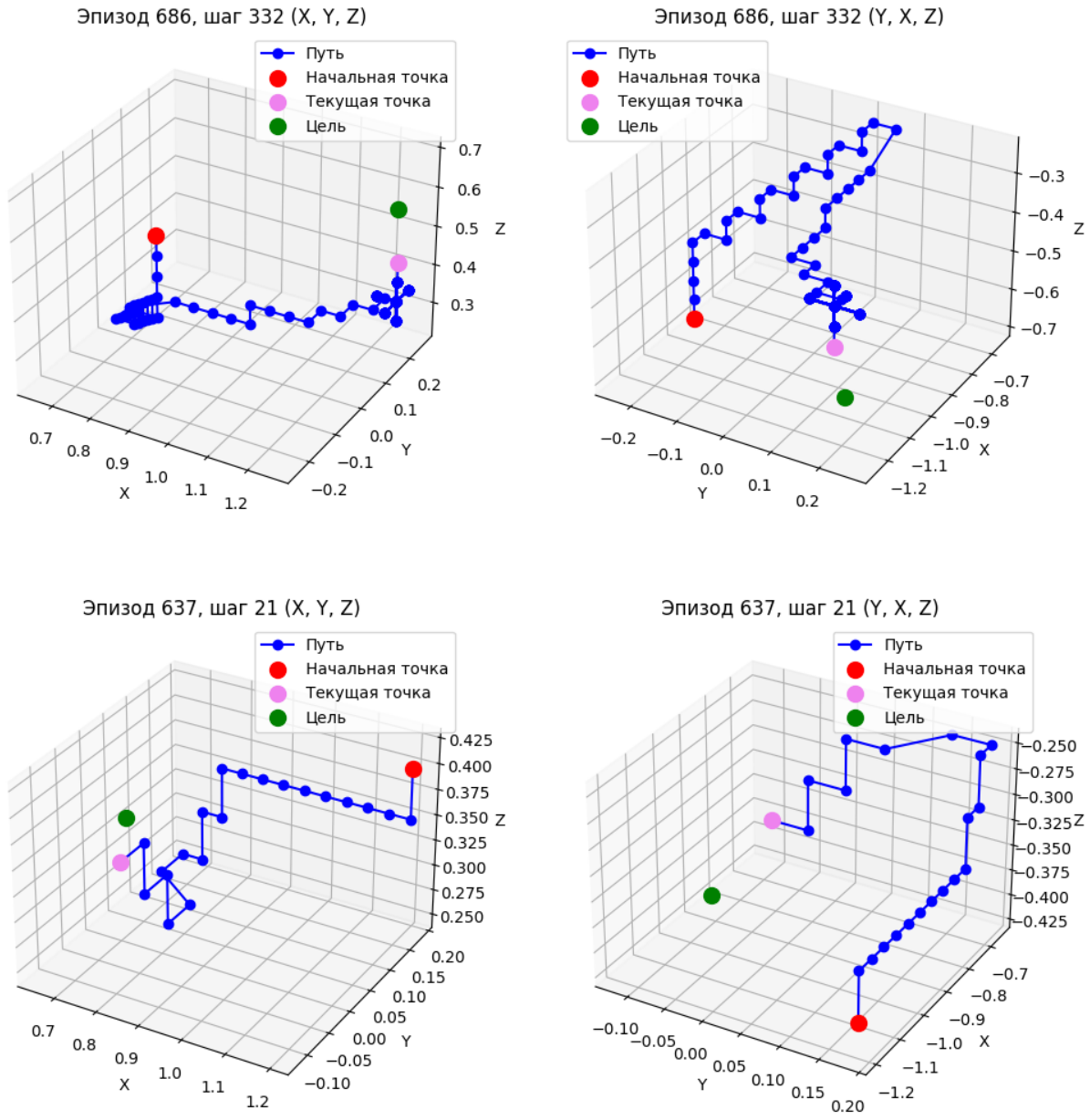


Рис. 36: Построение траектории на этапе обучения (конец обучения)

чивало гибкость и удобство при работе с моделями роботов. В качестве тестового объекта использовалась модель Universal Robots (UR5). Для визуализации экспериментов использовалась rViz.

4.1 Эксперименты, связанные с представлением препятствий

Препятствия были взяты из созданной библиотеки препятствий или получены из симулятора Gazebo. Также облако точек (PointCloud) было собрано в Isaac Sim, и предложенный метод был протестирован на нем.

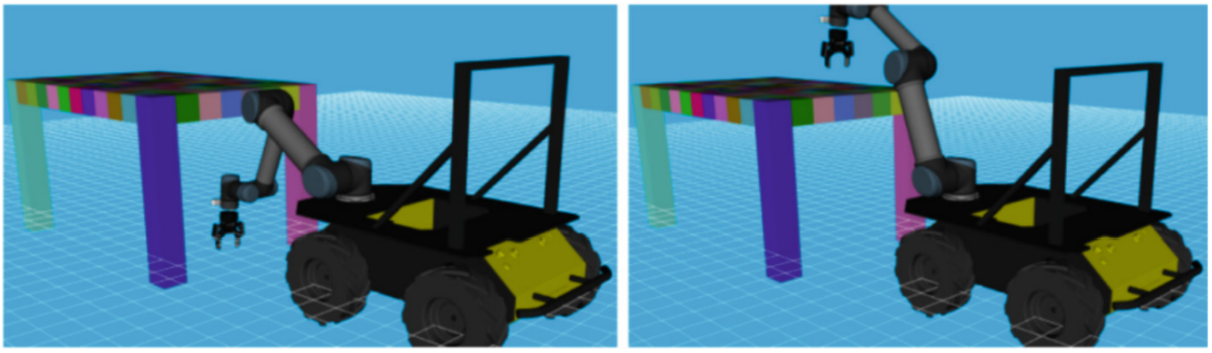


Рис. 37: Начальная и целевая позиции робота

Использовалось 2 сценария для отработки метода планирования. В первой сцене находится только стол, начальная позиция манипулятора робота под столом и направлена вниз. Необходимо построить траекторию так, чтобы захват находился над рабочей поверхностью стола. Иллюстрация начальной и целевой позиции робота показана на рисунке 37. В этом эксперименте поставили задачу планирования из позиции, где задача манипулятора близка к полу под столом (например, для того чтобы что-то взять).

На втором этапе есть шкаф. Работу необходимо спланировать между полками. Пример такого планирования показан на рисунке 38.

Эксперименты проводились при следующих условиях:

- размер куба составляет 5 и 10 см;
- кубы были объединены и не объединены;
- эксперименты проводились, когда начальная траектория, рассчитанная с помощью RRTConnect, использовалась в алгоритме TrajOpt, и когда использовался только TrajOpt.

Как показано в таблице 1, для одного тестового случая была построена одна траектория RRTConnect, которая была оптимизирована один раз в TrajOpt. Успех и время выполнения алгоритма TrajOpt измерялись как метрики (время, необходимое алгоритму для нахождения траектории или генерации ошибки). Если использовался RRTConnect, то результаты расчета алгоритма RRTConnect были поданы в TrajOpt. Если использовался только TrajOpt, то начальная траектория принималась как интерполированная траектория вдоль дуги от начальной позы захвата до целевой позы захвата. Длина начальной траектории в обоих случаях составила 11 шагов.

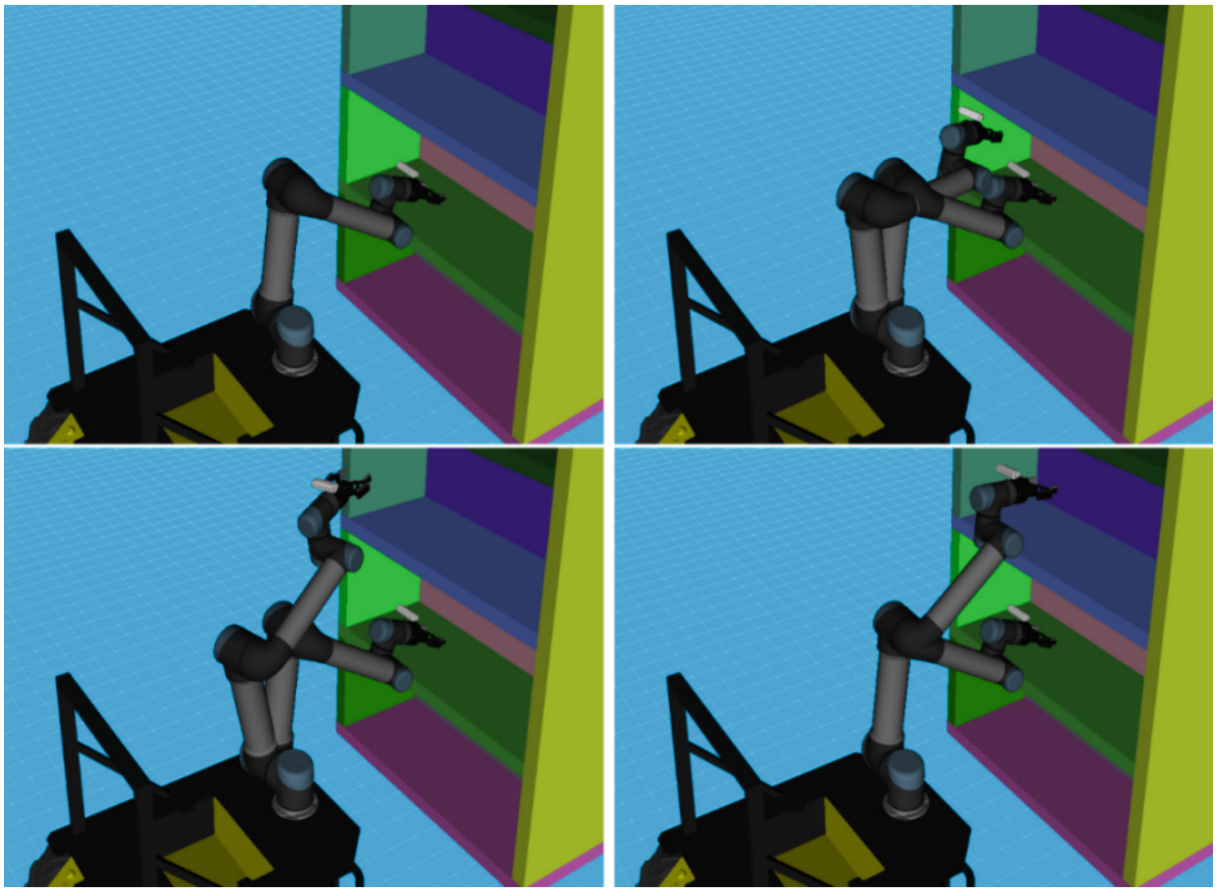


Рис. 38: Планирование между полками шкафа

В таблице 1 показаны результаты экспериментов. Время планирования в таблице указано только для TrajOpt. Как видно из таблицы, увеличение кубов Octomap позволяет уменьшить время планирования и увеличить вероятность успеха. Например, при планировании между полками шкафа, когда каждая полка представляла собой один параллелепипед, TrajOpt построил траекторию. А в аналогичной сцене с Octomap он смог построить траекторию только с помощью RRTConnect. Сравнение времени планирования показано на рисунке 39, а успешности показано на 2.

4.2 Эксперименты, связанные с генерацией начальной траектории

Для проведения экспериментов первый датасет был собран вручную с помощью визуализатора rViz. На нем производилось обучение и валидация результатов, которые были получение при помощи RNN. На сцену была добавлена платформа Husky с манипулятором UR5. Робот в визуализаторе rViz выставлялся в положение, которое визуально было выполнимо: отсутствовали пересечения с препятствиями и самокол-

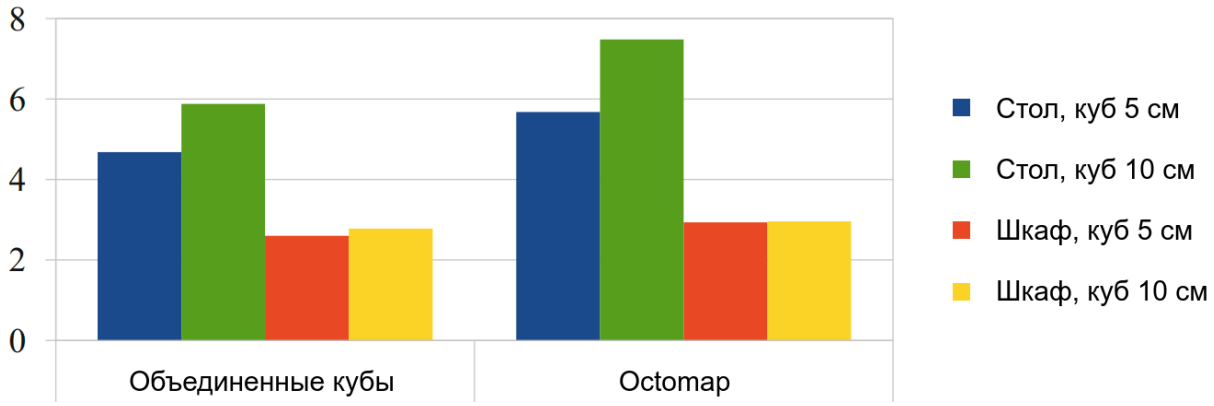


Рис. 39: Время планирования (сек)

лизии с роботом или мобильной платформой. И это положение сохранялось в файл. Таким образом было собрано 300 сценариев (по 100 для каждой из 3-х сцен). Каждый сценарий включает в себя начальное и целевое положение робота, а также описание препятствий на сцене.

С собранными сценариями был проведен эксперимент: на сцене без препятствий была запущена оптимизация силами TrajOpt. Сценарии, при запуске которых TrajOpt не смог найти траекторию, были удалены. Это говорит о том, что изначально были подобраны невыполнимые сценарии (или начальное/целевое положение робота не подходит для TrajOpt). Однако стоит отметить, что RRTConnect смог построить траектории во всех 300 сценариях. Поэтому можно сделать вывод, что TrajOpt по-другому оценивает достижимость того или иного положения робота.

После очищения датасета от проблемных положений, был проведен эксперимент с планированием: в TrajOpt передавалась траектория, сгенерированная RRTConnect. В итоге количество сценариев не изменилось: RRTConnect не портит результаты TrajOpt.

После очищения датасета получилось 218 сценариев, включая:

- для сцены со шкафом - 55 шт.;
- для сцены со стулом - 77 шт.;
- для сцены со столом - 86 шт.

При запуске экспериментов с планированием на сцене с препятствиями выяснилось, что TrajOpt может справляться с задачей без подачи в него начальной траектории. Если TrajOpt может оптимизировать траекторию без RRTConnect, то в датасет записывался этот результат. Если не может, то запускался RRTConnect в цикле (до

Таблица 1: Таблица с описанием экспериментов

Сцена эксперимента	Octomap, успех	Octomap, время, сек	Кубы об., успех	Кубы об., время, сек
Планирование из-под стола, куб 10 см, (RRTConnect и TrajOpt)	80%	7.48	80%	4.68
Планирование из-под стола, куб 10 см, (только TrajOpt)	0%	3.58	0%	2.66
Планирование из-под стола, куб 5 см, (RRTConnect и TrajOpt)	75%	5.68	80%	4.68
Планирование из-под стола, куб 5 см, (только TrajOpt)	0%	6.7	0%	2.15
Планирование между полками в шкафу, куб 10 см (RRTConnect и TrajOpt)	95%	2.97	100%	2.78
Планирование между полками в шкафу, куб 10 см (только TrajOpt)	0%	2.54	100%	1.36
Планирование между полками в шкафу, куб 5 см (RRTConnect и TrajOpt)	100%	2.94	100%	2.6
Планирование между полками в шкафу, куб 5 см (только TrajOpt)	0%	3.8	100%	1.29

Таблица 2: Успешные запуски

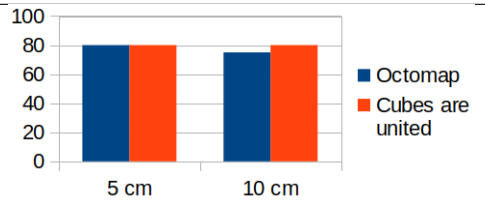
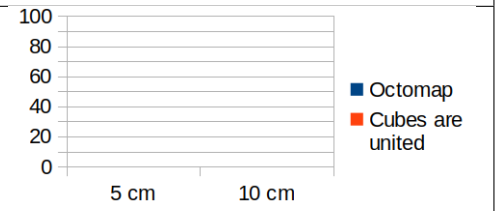
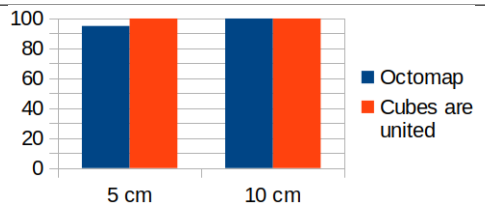
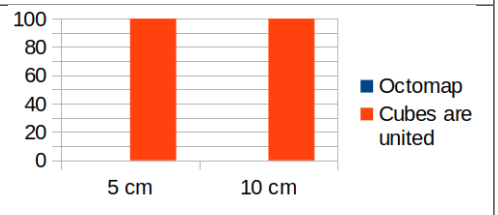
Сцена	RRTConnect + TrajOpt	Только TrajOpt
Стол		
Шкаф		

Таблица 3: Результаты экспериментов.

Эксперименты	Шкаф	Стул	Стол
TrajOpt без RRTConnect нашел путь	44 (80%)	64 (83%)	56 (65%)
TrajOpt + RRTConnect нашел путь с первого раза	8 (14%)	12 (16%)	20 (23%)
TrajOpt + RRTConnect нашел путь не с первого раза	1 (2%)	0	10 (12%)
TrajOpt + RRTConnect не нашел путь	2 (4%)	1 (0,01)	0
Итого	55	77	86

10 раз), т.к. алгоритм обладает стохастичностью и может при одинаковых входных данных давать разный результат.

Результаты экспериментов представлены в таблице 3.

Полученные данные использовались для обучения сети RNN. После обучения полученные результаты были проверены на валидационном датасете. Результаты работы разных подходов на этом датасете приведены в таблице 4 и на диаграмме 40.

При планировании только с помощью TrajOpt учитывалась одна попытка, т.к. алгоритм не имеет смысла запускать еще, если он с первого раза не нашел путь. Эксперименты, которые проводились, это подтвердили: вариативности результатов нет.

Если использовался алгоритм RRTConnect для генерации начальной траектории, то эксперименты по планированию с его участием проводились до 10 раз, т.к.

Таблица 4: Сравнение различных методов на валидационном датасете

Метод	Шкаф	Стул	Стол
Только TrajOpt с 1 попытки	14 (64%)	23 (77%)	15 (52%)
RRTConnect + TrajOpt с 1 попытки	7 (32%)	5 (17%)	10 (34%)
RRTConnect + TrajOpt не с 1 попытки	1 (4%)	2 (6%)	4 (14%)
RRTConnect + TrajOpt не нашел путь	0	0	0
RNN + TrajOpt с 1 попытки	22 (100%)	30 (100%)	29 (100%)

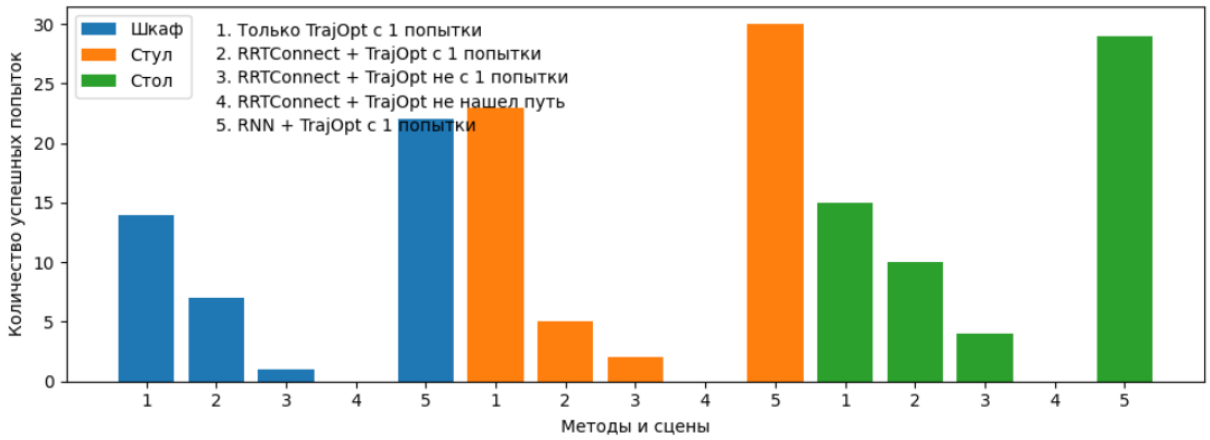


Рис. 40: Сравнение различных методов для объектов (шкаф, стул, стол)

алгоритм предлагает разные варианты решения задачи при одинаковых входных данных. Было принято решение, что 10 раз попробовать построить траекторию достаточно. На практике не встречалось, чтобы дальше 6 попытки были удачными. Обычно если RRTConnect мог предложить хорошую начальную траекторию, то он это делал в рамках 5 попыток.

Казалось бы, на небольшом датасете RNN получилось продемонстрировать отличные результаты: 100% предложенных траекторий оптимизировалось TrajOpt. Однако это ложные результаты: TrajOpt, несмотря на то, что с траекторией ему подается отдельно начальное и конечное положение, не учитывает их. Траектории, предложенные RNN, оказались «неправильными», т.к. не содержали точку начала и конца траектории. В связи с этим возникло много вопросов относительно корректной работы алгоритма TrajOpt, однако при детальном рассмотрении траектории от RNN все вопросы были сняты.

Сгенерированный датасет

Поскольку собирать датасет вручную слишком долго и его качество оставляет

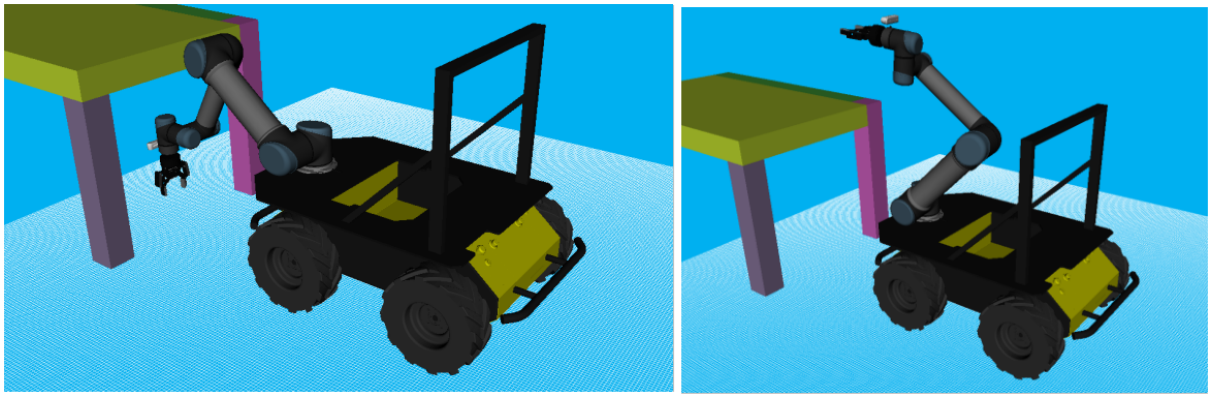


Рис. 41: Начальное (слева) и целевое (справа) положения робота

желать лучшего (треть сценариев пришлось удалить), было принято решение сгенерировать датасет больше.

Для этого:

- было принято решение целевое положение робота представить в декартовых координатах рабочего инструмента;
- были определены границы приблизительной зоны досягаемости робота по осям x, y, z (взяты примерно по результатам исследования модели робота-манипулятора в rViz), были установлены границы для углов поворота Эйлера (roll, pitch, yaw), как показано на рисунке 42;
- был определен шаг 0.1 метра для перемещения по осям x, y, z и 0.1 радиана для осей вращения (в основном по roll и yaw), при получении большого количества вариантов они прореживались, чтобы итоговое количество поз получилось не слишком большим;
- простым перебором сгенерированы разные комбинации (всего получилось 112 320 позы);
- все предложенные позы проверены на наличие решения задачи обратной кинематики (осталось 87 049 поз);
- определено, что 100 положений будет достаточно для проведения эксперимента (группируем первые 50 со вторыми 50, и получаем итого 2 500 вариантов);
- полученные решения были записаны и перемешаны между собой, а затем проитерированы по порядку и проверены на самоколлизию (как только набралось оптимальное количество подходящих положений - 100, завершили перебор);

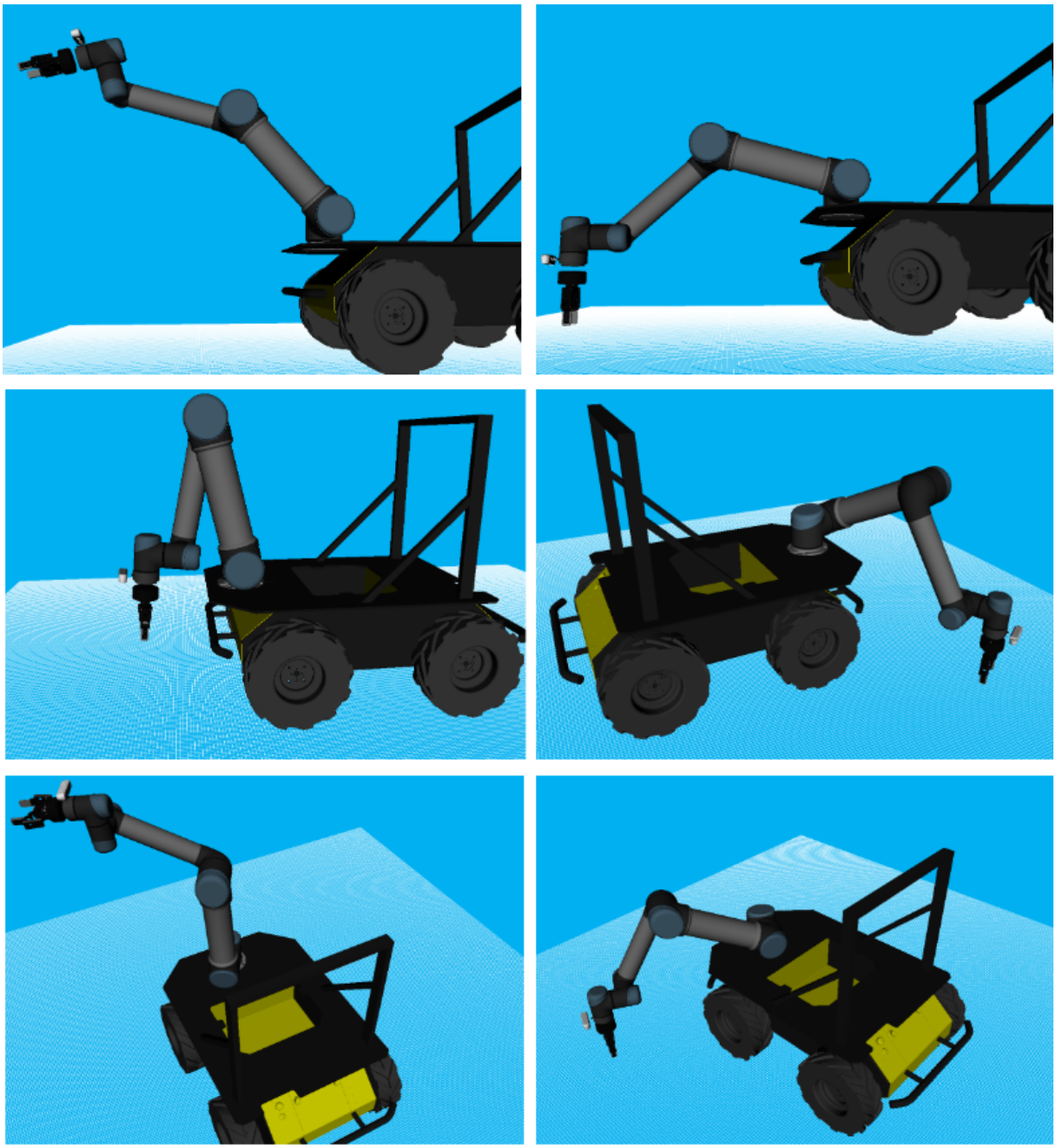


Рис. 42: Определение границ в пространстве для сбора датасета

- все положения перегруппировали между собой, первое положение - стартвое, второе - целевое;
- на полученных парах проверили, может ли TrajOpt выполнить траекторию на пустой сцене (в 2 374 случаях смог - эти данные оставляем для обучения, в 126 случаях не смог - эти данные нам не нужны).

После того как получили данные, с которыми теперь можно работать, провели точно такие же эксперименты, что были сделаны на датасете, собранном вручную.

Таблица 5: Результаты экспериментов по сбору датасета.

Эксперименты	Шкаф	Стул	Стол
TrajOpt без RRTConnect нашел путь	687 (96,35%)	912 (87,20%)	1063 (88,6%)
TrajOpt + RRTConnect нашел путь с первого раза	25 (0,04%)	123 (11,76%)	104 (8,67%)
TrajOpt + RRTConnect нашел путь не с первого раза	1 (0%)	11 (0.01%)	33 (0,02%)
TrajOpt + RRTConnect не нашел путь	0 (0%)	0 (0%)	0 (0%)
Итого	713	1046	1200

Единственное отличие состоит в том, что все траектории запускаем на каждой сцене, поэтому в код перед началом планирования была добавлена проверка на самоколлизию и коллизию с препятствиями на сцене. Если коллизия есть изначально, то такое положение робота нам не интересно: TrajOpt гарантированно не сможет оптимизировать траекторию, если робот в исходном или целевом положении пересекает сам себя или препятствие.

На 3-х разных сценах с препятствиями было проверено 2 374 сценариев с одними и теми же данными для начального и целевого положения робота. Результаты экспериментов представлены в таблице 6.

Собранные данные были поделены на обучающий (90%) и валидационный датасет (10%). Получилась длина датасета:

- для сцены со шкафом - 642 для обучения и 71 для валидации;
- для сцены со стулом - 941 для обучения и 105 для валидации;
- для сцены со столом - 1080 для обучения и 120 для валидации.

Итого для обучения используется 2 663 положения, для валидации - 296.

Как видно по результатам, траектории проходят таким образом, что в основном TrajOpt может их рассчитать самостоятельно (см. таблицу 6). Это говорит о том, что начальное и целевое положение робота-манипулятора расположено на расстоянии от препятствий. И обучение и проверка на валидационном датасете тому подтверждение.

Таблица 6: Результаты на валидационном датасете

Эксперименты	Все сцены
TrajOpt без RRTConnect нашел путь	658 (74%)
TrajOpt + RRTConnect нашел путь с первого раза	0 (0%)
TrajOpt + RRTConnect не нашел путь с первого раза	232 (0,26%)
Среднее время планирования при успешном решении	0,89 сек.
Всего сценариев (для 3х сцен)	888
RNN + RRTConnect нашел путь с первого раза	888 (100%)
Среднее время с RNN планирования при успешном решении	0,42 сек.

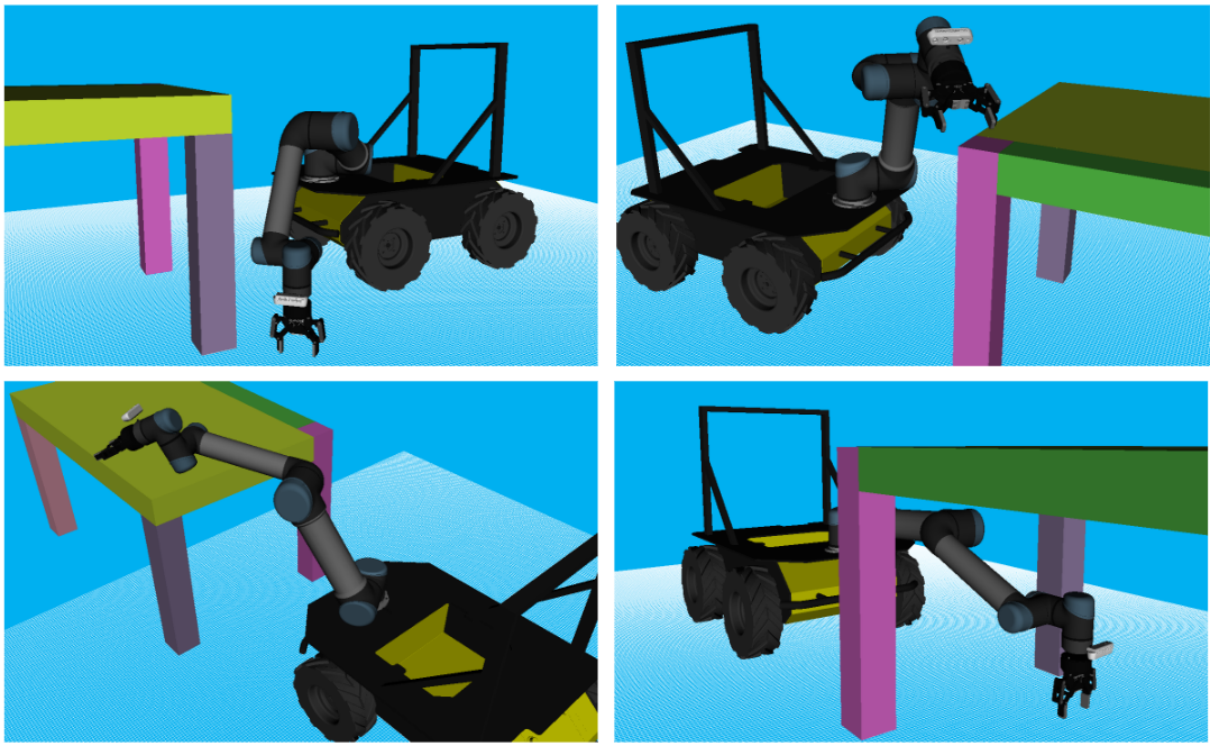


Рис. 43: Определение границ в пространстве для сбора датасета для сцены со столом

Поэтому было принято решение собрать свой датасет для каждого вида сцены, определив позиции робота как можно ближе к препятствию. Таким образом, для каждой сцены были найдены свои границы. Границы для сцены со столом приведены на рисунке 43.

Параметры датасета с препятствием:

- препятствие на сцене - стол;

Таблица 7: Результаты на валидационном датасете

Эксперименты	Сцена со столом
TrajOpt без RRTConnect нашел путь	46 (55%)
TrajOpt + RRTConnect нашел путь с первого раза	25 (30%)
TrajOpt + RRTConnect нашел путь не с первого раза	13 (15%)
Среднее время планирования при успешном решении (TrajOpt и TrajOpt + RRTConnect в случае успеха)	4,19 сек.
TrajOpt + RRTConnect не нашел путь	0 (0%)
RNN + RRTConnect нашел путь с первого раза	20 (24%)
Среднее время с RNN планирования при успешном решении	0,5 сек.

- границы для генерации позы находятся близко к краю стола;
- количество возможных положений - 51 840 шт.;
- из них только для 37 154 нашлось решение обратной задачи кинематики;
- положения перемешаны и проверены на наличие коллизий, как только алгоритм набрал 61 положение для генерации 930 траекторий, он остановился (при этом на каждое достижимое положение робота приходилось 3 недостижимых из-за коллизии с препятствием);
- из 930 траекторий на сцене при отсутствии препятствий достижимыми оказались 843, недостижимыми - 87;
- из 843 сценариев 10% было отложено на валидационный датасет, остальные 90% использовались для обучения.

В итоге датасет был сгенерирован в непосредственной близости робота от препятствий (759 положений получилось для обучения и 84 для валидации). Результаты работы разных подходов планирования приведены в таблице 7.

Пример сгенерированной траектории показан на рисунке 44. Первые 2 больших изображения - это поданные на вход положения робота (начальное и целевое). Всего в траектории было 20 положений.

С учетом всех вышеописанных ситуаций было принято решение подготовить такой датасет, в котором будут положения начала и конца траектории такие, что

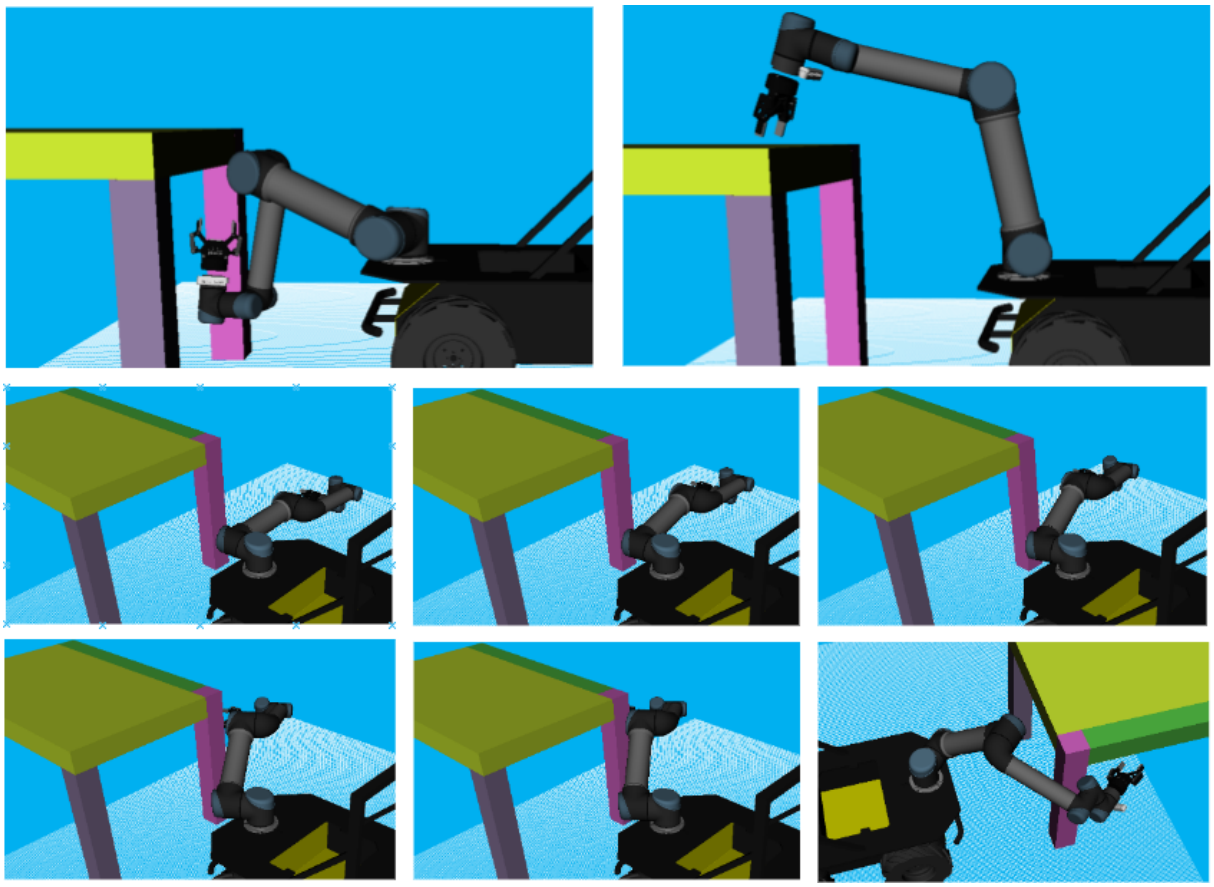


Рис. 44: Начальное и целевое положение (сверху), и положения из сгенерированной траектории (снизу)

TrajOpt между ними не сможет построить траекторию. Результаты применения всех алгоритмов представлены в таблице 8.

На основе проведенных экспериментов и анализа полученных результатов можно сделать несколько ключевых выводов относительно эффективности различных подходов к планированию траекторий манипулятора в сложных условиях.

Наиболее успешным методом оказался TrajOpt в комбинации с RRTConnect, который в 65% случаев смог найти путь с первого раза, а в целом количество успешных попыток (включая повторные попытки) составило 99%. Это подтверждает высокую эффективность данного метода в условиях, когда требуется решение задач с учетом значительных препятствий.

Алгоритм TrajOpt без RRTConnect продемонстрировал значительно худшие результаты, не найдя ни одного пути (0%). В то время как методы, основанные на RNN и RL, оказались более успешными; RNN смог найти путь в 24% случаев с первого раза, а RL — в 82%. Это говорит о том, что интеграция методов машинного обучения

Таблица 8: Результаты всех подходов на валидационном датасете

Эксперименты	Сцена со столом
TrajOpt без RRTConnect нашел путь	0 (0%)
TrajOpt + RRTConnect нашел путь с первого раза	60 (65%)
TrajOpt + RRTConnect нашел путь не с первого раза	31 (34%)
TrajOpt + RRTConnect не нашел путь (лимит 10 попыток)	1 (1%)
Среднее время планирования при успешном решении (TrajOpt и TrajOpt + RRTConnect в случае успеха)	8,19 сек.
TrajOpt + RRTConnect не нашел путь	67 (74%)
TrajOpt + RRN нашел путь с первого раза	20 (24%)
Среднее время с RNN планирования при успешном решении	1,9 сек.
TrajOpt + RL не нашел путь	16 (18%)
TrajOpt + RL нашел путь с первого раза	74 (82%)
Среднее время с RL планирования при успешном решении	7,5 сек.

и оптимизации значительно повышает возможности планирования.

Следует обратить внимание на среднее время планирования при успешном решении. Наименее затратным оказался метод с использованием RNN, с средним временем 1,9 секунды. Однако этот метод ненадежен. В то время как методы TrajOpt с RRTConnect и RL показали средние времена 8,19 и 7,5 секунды соответственно, что указывает на более высокую вычислительную сложность при использовании этих стратегий.

Значительная доля неудачных попыток (74% для TrajOpt + RRTConnect) указывает на необходимость дальнейшей оптимизации методов для улучшения их устойчивости в сложных сценариях. Это подчеркивает важность доработки существующих алгоритмов и изучения новых подходов для достижения более надежных результатов в реальных условиях.

В итоге, исследования подтвердили многообещающий характер комбинации методов, что открывает новые горизонты для дальнейших разработок в области планирования траекторий манипуляторов, особенно в окружающей среде с множеством препятствий. Необходимо продолжать исследование и оптимизацию различных стратегий, чтобы обеспечивать максимальную эффективность и надежность в будущем.

5 Заключение

В ходе проведенного исследования были достигнуты несколько значительных результатов, которые подчеркивают эффективность предложенных методов планирования траекторий для роботизированных манипуляторов.

Основные результаты работы

- Разработан новый подход к планированию траекторий, который сочетает алгоритм TrajOpt с методами глубокого обучения. Это позволило повысить эффективность и точность построения траекторий в сложных условиях. Наиболее удачным решение оказалось применение RL-подхода.

- В рамках работы были созданы и реализованы алгоритмы, позволяющие улучшить обработку данных с сенсоров. Данные алгоритмы динамически корректируют траектории манипулятора в реальном времени, что обеспечивает большую гибкость в управлении.

- Проведенные вычислительные эксперименты подтвердили гипотезу о том, что использование нейронных сетей в сочетании с традиционными методами планирования значительно улучшает результаты выполнения задач. Эксперименты показали сокращение времени на расчет траекторий и увеличение их надежности в условиях изменения обстановки.

- Обсуждены перспективы применения новых подходов для коллаборативных роботов. Исследование показало, что адаптация алгоритмов TrajOpt и CHOMP с учетом предсказания поведения людей способствует повышению эффективности взаимодействия между роботами и людьми в рабочих условиях.

В заключение, результаты нашего исследования подчеркивают высокую значимость внедрения новых методов в практику планирования траекторий для роботизированных систем. Они не только повышают качество и безопасность работы манипуляторов, но и открывают новые направления для исследований в области робототехники.

Одним из перспективных направлений является дальнейшее углубленное изучение методов обучения с подкреплением для повышения адаптивности роботизированных систем к изменениям в окружающей среде. Также стоит рассмотреть возмож-

ность расширения разработанных алгоритмов к более сложным автоматизированным системам, что позволит значительно улучшить их производительность и внедрение в различные сферы применения.

Список литературы

1. *Hart P. E.*, *Nilsson N. J.*, *Raphael B.* A formal basis for the heuristic determination of minimum cost paths // IEEE transactions on Systems Science and Cybernetics. — 1968. — Т. 4, № 2. — С. 100—107.
2. *Gilbert E Johnson D K. S.* A Fast Procedure for Computing the Distance between Complex Objects in ThreeDimensional Space // IEEE Journal of Robotics and Automation. — 1988. — Т. 4, № 2. — С. 193—203.
3. *Hadwiger H.* Minkowskische Addition und Subtraktion beliebiger Punktmengen und die Theoreme von Erhard Schmidt // Math. Z. — 1988. — Т. 53, № 3. — С. 210—218.
4. *Kavraki L.*, *Latombe J.-C.* Randomized preprocessing of configuration space for path planning: articulated robots // Proceedings of IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS'94). Т. 3. — 1994. — 1764—1771 vol.3. — DOI: 10.1109/IROS.1994.407619.
5. *LaValle S.*, *Kuffner J.* Randomized kinodynamic planning // Proceedings 1999 IEEE International Conference on Robotics and Automation (Cat. No.99CH36288C). Т. 1. — 1999. — 473—479 vol.1. — DOI: 10.1109/ROBOT.1999.770022.
6. *Kuffner J.*, *LaValle S.* RRT-connect: An efficient approach to single-query path planning // Proceedings 2000 ICRA. Millennium Conference. IEEE International Conference on Robotics and Automation. Symposia Proceedings (Cat. No.00CH37065). Т. 2. — 2000. — 995—1001 vol.2. — DOI: 10.1109/ROBOT.2000.844730.
7. *Bergen G. van den.* Proximity Queries and Penetration Depth Computation on 3D Game Objects // In Game Developers Conference(GDC). — 2001.
8. CHOMP: Gradient optimization techniques for efficient motion planning / N. Ratliff, M. Zucker, J. A. Bagnell, S. Srinivasa // 2009 IEEE International Conference on Robotics and Automation. — 2009. — С. 489—494. — DOI: 10.1109/ROBOT.2009.5152817.
9. Motion planning with sequential convex optimization and convex collision checking / J. Schulman, Y. Duan, J. Ho, A. X. Lee, I. Awwal, H. Bradlow, J. Pan, S. Patil, K. Goldberg, P. Abbeel // The International Journal of Robotics Research. — 2014. — Т. 33. — С. 1251—1270.

10. *Carlos Quintero-Pena Constantinos Chamzas V. U.* Motion Planning via Bayesian Learning in the Dark // Workshop on Machine Learning for Motion Planning ICRA 2021. — 2021.

11. *Carlos Quintero-Peña Anastasios Kyrillidis L. E. K.* Robust Optimization-based Motion Planning for high-DOF Robots under Sensing Uncertainty // IEEE International Conference on Robotics and Automation (ICRA). — 2021.

12. *Feng Yichang Zhang Haiyun W. J.* Adaptive Gaussian Process based Stochastic Trajectory Optimization for Motion Planning // arXiv:2112.14953v1. — 2021.

13. *Amber Xie Youngwoon Lee P. A.* Language-Conditioned Path Planning // Conference on Robot Learning. — 2023.

14. *Feng Yichang Wang Jin L. G.* Optimal Motion Planning using Finite Fourier Series in a Learning-based Collision Field // -. — 2023.

15. *Jiahe Pan Kerry He J. M. O.* Variable Grasp Pose and Commitment for Trajectory Optimization // arXiv:2305.12455v1. — 2023.

16. *Julius Jankowski Lara Brudermüller N. H.* VP-STO: Via-point-based Stochastic Trajectory Optimization for Reactive Robot Behavior // arXiv:2210.04067v2. — 2023.

17. *Minsung Yoon Mincheul Kang D. P.* Learning-based Initialization of Trajectory Optimization for Path-following Problems of Redundant Manipulators // IEEE International Conference on Robotics and Automation (ICRA). — 2023.

18. *Yeseung Kim Jinwoo Kim D. P.* GraphDistNet: A Graph-based Collision-distance Estimator for Gradient-based Trajectory Optimization // IEEE Robotics and Automation Letters. — 2023.

19. *Chenlin Ming Zitong Wang B. Z.* Stochastic Trajectory Optimization for Demonstration Imitation // arXiv:2408.03131v2. — 2024.

20. *David von Wrangel R. T.* Using Graphs of Convex Sets to Guide Nonconvex Trajectory Optimization // -. — 2024.

21. *Eliot Xing Vernon Luk J. O.* Stabilizing Reinforcement Learning In Differentiable Multiphysics Simulation // arXiv:2410.12172v2. — 2024.

22. *Eliot Xing Vernon Luk J. O.* Stabilizing Reinforcement Learning in Differentiable Simulation of Deformables // arXiv:2412.12089v1. — 2024. — DOI: http://dx.doi.org/10.48550/arXiv.2412.12089.

23. *Fan Yang Wenxuan Zhou Z. L.* Reinforcement Learning in a Safety-Embedded MDP with Trajectory Optimization // arXiv:2310.06903v2. — 2024.

24. *Guangyao Zhou Sivaramakrishnan Swaminathan R. V. R.* Diffusion Model Predictive Control // arXiv:2410.05364v1. — 2024.

25. *Bo-Hsun Chen P.-C. L.* Deep Imitation Learning for Optimal Trajectory Planning and Initial Condition Optimization for an Unstable Dynamic System // Advanced Intelligent Systems. — 2024. — DOI: 10.1002/aisy.202300379.

26. *Huy Hoang Nguyen Minh Nhat Vu F. B.* Language-Driven Closed-Loop Grasping with Model-Predictive Trajectory Replanning // arXiv:2406.09039v3. — 2024.

27. *João Carvalho An T. Le P. K.* Motion Planning Diffusion: Learning and Adapting Robot Motion Planning with Diffusion Models // arXiv:2412.19948v1. — 2024.

28. *Jonathan Michaux Seth Isaacson C. E. A.* Let's Make a Splan: Risk-Aware Trajectory Optimization in a Normalized Gaussian Splat // arXiv:2409.16915v1. — 2024.

29. *Jonathan Michaux Adam Li Q. C.* Safe Planning for Articulated Robots Using Reachability-based Obstacle Avoidance With Spheres // -. — 2024.

30. *Ya-Jun Pan Qiguang Chen N. C.* Survey on Recent Advances in Planning and Control for Collaborative Robotics // IEEJ Journal of Industry Applications. — 2024. — № 2. — С. 139—151.

31. *Jun Yamada Shaohong Zhong J. C.* D-cubed: Latent Diffusion Trajectory Optimisation For Dexterous Deformable Menipulation // arXiv:2403.12861v1. — 2024.

32. *Kostas E. Bekris Joe Doerr P. M.* The State of Robot Motion Generation // arXiv:2410.12172v2. — 2024.

33. *Mingyo Seo Yoonyoung Cho Y. S.* PRESTO: Fast motion planning using diffusion models based on key-configuration environment representation // arXiv:2409.16012v1. — 2024.

34. *Ramkumar Natarajan Shohin Mukherjee H. C.* PINSAT: Parallelized Interleaving of Graph Search and Trajectory Optimization for Kinodynamic Motion Planning // arXiv:2401.08948v2. — 2024.

35. *Sankaranarayanan Natarajan P. F. K.* ITPNet: Initial Trajectory Prediction Network - A deep learning framework for predicting initial trajectories to warm start optimization-based motion planners // -. — 2024. — DOI: https://doi.org/10.21203/rs.3.rs-4110511/v1.

36. *Thomas Power D. B.* Constrained Stein Variational Trajectory Optimization // arXiv:2308.12110v3. — 2024.

37. *Yeping Wang Carter Sifferman M. G.* IKLink: End-Effector Trajectory Tracking with Minimal Reconfigurations // arXiv:2402.16154v2. — 2024.

38. *Yongseok Kwon Jonathan Michaux S. I.* Conformalized Reachable Sets for Obstacle Avoidance With Spheres // arXiv:2410.09924v1. — 2024.

39. *Yu Xiang Sai Haneesh Allu R. P.* Grasping Trajectory Optimization with Point Clouds // arXiv:2403.05466v2. — 2024.

40. *Yulin Li Chunxin Zheng K. C.* Collision-Free Trajectory Optimization in Cluttered Environments Using Sums-of-Squares Programming // arXiv:2404.05242v2. — 2024.

41. *Zhengtong Xu Y. S.* LeTO: Learning Constrained Visuomotor Policy with Differentiable Trajectory Optimization // arXiv:2401.17500v3. — 2024.

42. *authors A.* Reflect-Then-Plan: Offline Model-based Planning Through A Doubly Bayesian Lens // Paper at ICLR 2025. — 2025.

43. *Yeping Wang M. G.* Anytime Planning for End-Effector Trajectory Tracking // arXiv:2502.03676v1. — 2025.