Разработка методов коррекции траектории движения робототехнического манипулятора

Рублева Елена Алексеевна

Центр когнитивного моделирования МФТИ

2025 - Научно-исследовательский семинар

Мотивация

Существующие методы планирования траектории движения робота-манипулятора плохо справляются с задачей:

- Не могут найти траекторию за оптимальное время;
- Не могут решить задачу планирования, если манипулятор пересекает сам себя или начальная траектория пересекает препятствия;
- Имеют большую вычислительную сложность;
- Не могут учитывать изменение среды (в частности расположение препятствий).

Предлагается разработать метод, который будет корректировать траекторию робота в зависимости от меняющихся условий среды.
Доработка метода направлена на:

- модификацию представления препятствий;
- поиск начального приближения (начальной догадки) траектории.

Постановка задачи

Дано: текущее положение манипулятора, целевое положение манипулятора, данные с сенсоров (камеры, лидар).

Нужно:

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 нужно:

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).

Формальная постановка задачи

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

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, $M = \text{агрегация}_{Z,Y,X}\left(\{\text{куб}(p, s(\|p - p_{\text{init}}\|)) \mid p \in \{p \in C \mid I(p_x, p_y, p_z) = 1\}\}\right)$, $s(d) = s_{\min} + k(d - d_{\min})$, $s_{\min}$ - 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.

Используемый метод - алгоритм TrajOpt (численная оптимизация)

TrajOpt [1] - это алгоритм планирования траектории, который оптимизирует траектории движения робота, чтобы избежать препятствий.

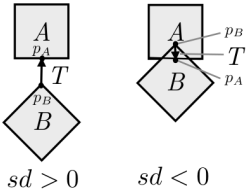


Рис. 1: Минимальное расстояние

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

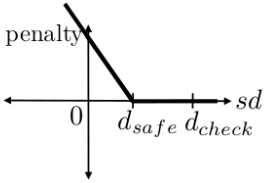


Рис. 2: Штраф за столкновение

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

- A and B - объекты,
- T - дистанция между объектами,
- sd - расстояние со знаком,
- θ – временной шаг,
- $\hat{n}$ - нормаль вектора,
- P_A and P_B – точки контакта между объектами.

[1] Schulman, J., et al.: Motion planning with equential convex optimization and convex collision checking. Int. J. Robot. Res. 33(9), 1251–1270 (2014)

Чтобы рассчитать штраф за столкновение, используется:

- Gilbert-Johnson-Kirti algorithm (GJK) для определения факта столкновения звена робота с препятствием;
- Алгоритм расширяющегося многогранника (EPA) для определения глубины проникновения.

$$d(A,B) = min\{\|x\| : x \in A - B\}, \text{ on condition } A \cap B \neq \emptyset \equiv 0 \in A - B$$

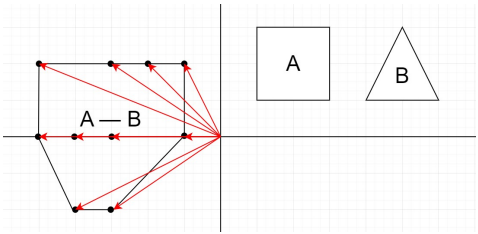


Рис. 3: Фигуры не пересекаются

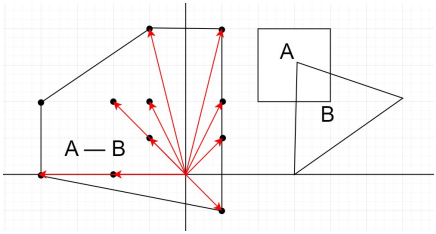


Рис. 4: Фигуры пересекаются

Улучшения алгоритма

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.

Базовое решение

В качестве базового решения для планирования мы использовали алгоритмы RRTConnect и TrajOpt. Реализация RRTConnect взята из MoveIt!. RRTConnect учитывает препятствия.

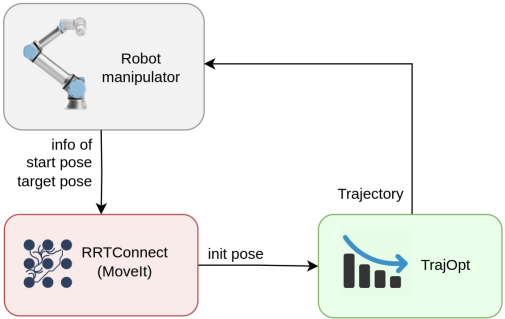


Рис. 5: Инструменты для планирования и оптимизации траектории

Программная реализация

Программная реализация на текущий момент состоит из следующих модулей:

- модуль с реализацией алгоритма TrajOpt (включает сторонние библиотеки Tesseract Robotics);
- модуль с моделью робота (робот-манипулятор UR5, подвижная платформа Husky, гриппер Robotiq);
- модуль с реализацией управления роботом-манипулятором с помощью TrajOpt (узлы ROS);
- модуль для реализации прямой и обратной кинематики[1] (8 решений с выбором наилучшего варианта);
- модуль симуляции и визуализации (Gazebo и rViz);
- модуль для обучения нейросети;
- модуль для запуска экспериментов;
- модель для укрупнения препятствий.

[1] Andersen R. S. Kinematics of a UR5. — 2018

Работа с препятствиями: Представление препятствий

Для представления препятствий:

- получаем данные с сенсоров как PointCloud;
- отфильтровываем данные с сенсоров вне досягаемости робота-манипулятора;
- преобразуем PointCloud в Octomap и объединяем кубы в параллелепипеды.

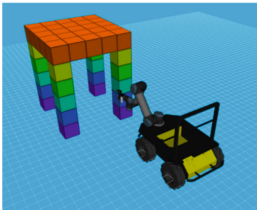


Рис. 6: Octomap

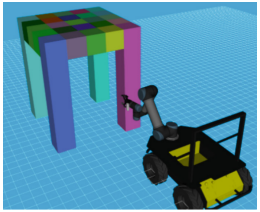


Рис. 7: по Z

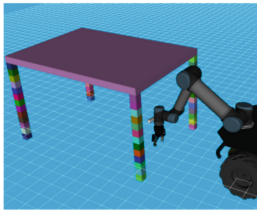


Рис. 8: по X, Y

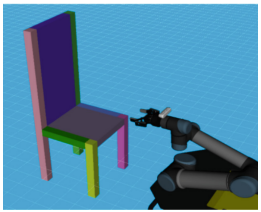


Рис. 9: по X, Y, Z

Работа с препятствиями: Gazebo симулятор

Пример сцены в симуляторе Gazebo.



Рис. 10: Сцена



Рис. 11: Данные с камеры

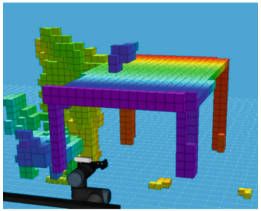


Рис. 12: Octomap

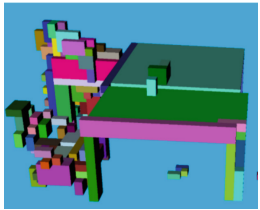


Рис. 13: Объединенные кубы по X, Y, Z

Работа с препятствиями: Библиотека

Для проведения экспериментов была разработана библиотека препятствий, в которую вошли такие препятствия, как: стол, стул, корзина, стена, шкаф.

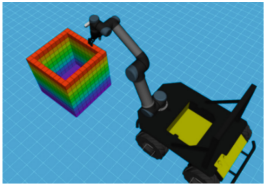


Рис. 14: Корзина

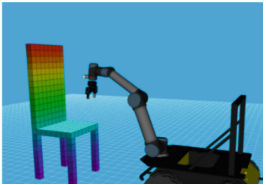


Рис. 15: Стул

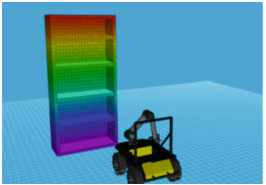


Рис. 16: Шкаф

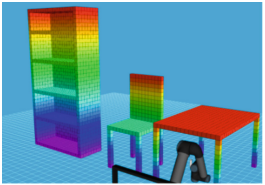


Рис. 17: Разные объекты

Работа с препятствиями: Эксперименты

Для экспериментов используется 2 сценария:

1. Только стол, исходное положение робота-захватчика находится под столом и направлено в пол;
2. Робот должен быть установлен между полками в шкафу.

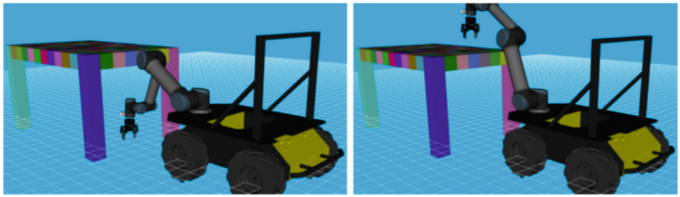


Рис. 18: Начальное и целевое положение робота

Работа с препятствиями: суть экспериментов

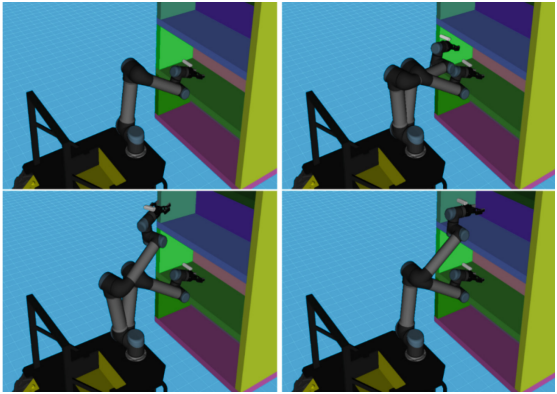


Рис. 19: Планирование между полками шкафа

Параметры эксперимента:

- модель робота - Universal Robots (UR5);
- ПО - ROS, Gazego, MoveIt!;
- алгоритмы - RRTConnect + TrajOpt, TrajOpt only;
- 2 сценария (со столом и со шкафом);
- размер кубов - 5 и 10 см;
- кубы объединены и не объединены;
- 20 сценариев для каждого параметра.

Работа с препятствиями: Результаты (время)

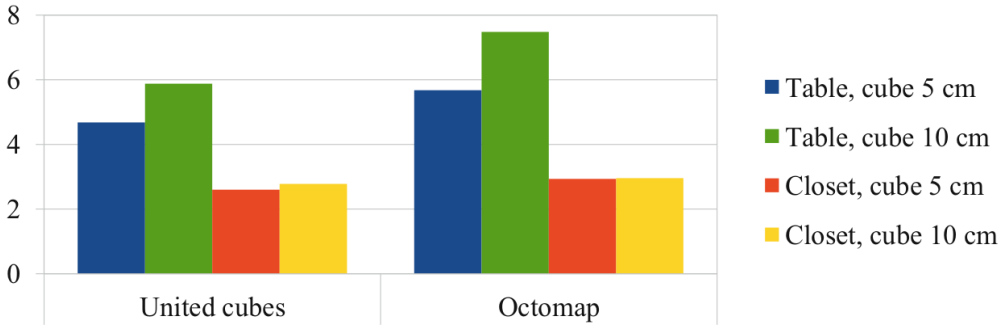
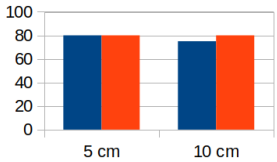
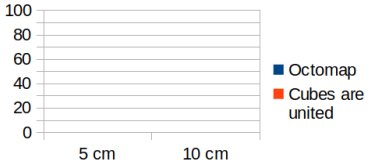
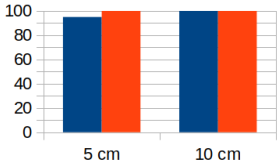
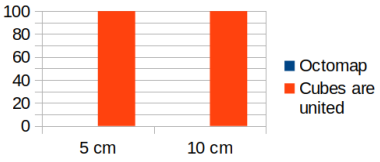


Рис. 20: Планирование, сек.

Работа с препятствиями: Результаты (success rate)

Type	RRTConnect + TrajOpt	TrajOpt only
Scene: table		
Scene: closet		

Генерация начальной догадки

Baseline (классическое решение) - **RRTConnect**. Baseline (нейросетевой подход) - **рекуррентная сеть**. В результате создаётся и обучается RNN с LSTM для обработки последовательных данных.
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.

Обучалось на 50 эпохах, размер батча 32. Метрика MAE для оценки точности при обучении.
Данные для обучения:

- сцена в виде octomap;
- начальное и целевое положение робота (6 джоинтов);
- траектория, сгенерированная с помощью RRTConnect;
- результат оптимизации TrajOpt.

Сцены для сбора данных для обучения RNN

Данные для обучения RNN собирались на 3-х сценах. В датасет включены данные с траекториями от RRTConnect, которые TrajOpt мог оптимизировать и не мог оптимизировать.

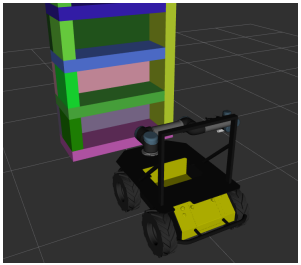


Рис. 21: Шкаф

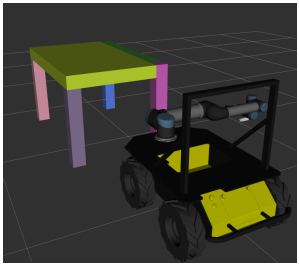


Рис. 22: Стол

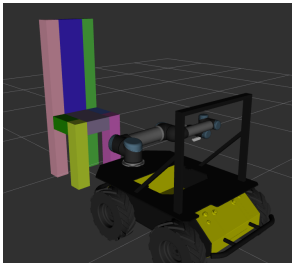


Рис. 23: Стул

Что было сделано

- Пересобран датасет для обучения модели (подобраны более разнообразные позы для робота, добавлены дополнительные поля, итоговый размер датасета составил 3350 строк);
- Написан код для автоматизации работы с датасетом;
- Переписан код модели RNN (в обучение добавлены дополнительные поля);
- Модель обучена на собранном датасете.

Результаты работы RNN

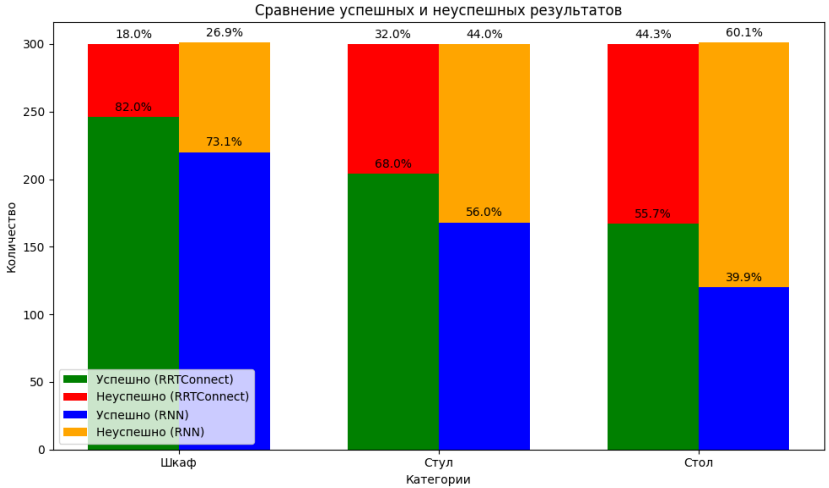


Рис. 24: Результаты экспериментов

Выводы

- Были проведены эксперименты в ROS с моделью UR5, исследуя различные сценарии, включая планирование под столами и между полками в шкафу, и показали, что большие кубы Octomap увеличили время планирования и показатели успешности.
- Было продемонстрировано, что объединение кубов в параллелепипеды значительно повышает эффективность планирования.
- Предложен нейросетевой подход для генерации начальной траектории робота-манипулятора.
- Дальнейшие планы: сравнить результаты обученной модели RNN с классическими методами (RRTConnect), добавить другую нейросетевую модель для оценки результатов RNN, провести эксперименты на роботе.

Спасибо за внимание!

rubleva.ea@phystech.edu