Московский государственный университет имени М.В. Ломоносова

Факультет вычислительной математики и кибернетики

Кафедра системного анализа

Выпускная квалификационная работа

«Повышение эффективности алгоритма KRRT* за счет использования внешних оценок множеств достижимости»

Студент 415 группы

Н. Ю. Заварзин

Научный руководитель

к.ф.-м.н., доцент П. А. Точилин

Москва, 2024

Содержание

1 Введение

Планирование движения — фундаментальная задача робототехники, методы решения которой обеспечивают работоспособность роботизированных систем в огромном числе отраслей: от сельского хозяйства до промышленного производства, от автономной городской навигации до здравоохранения.

Неформально говоря, задача планирования состоит в том, чтобы найти траекторию, которая переводила бы нашу систему из начальной точки в конечное множество, избегая при этом препятствий и минимизируя возможные затраты.

Ограничения, накладываемые на систему, как правило, разделяют на три типа.

1. Геометрические: обязывают нас искать решение, лежащее в некотором заданном множестве. Например, решение, не выходящее за пределы куба с центром в начальной точке.

2. Кинематические: при построении пути требуют, чтобы тот удовлетворял кинематическим связям рассматриваемой физической системы. Сюда относятся движение без проскальзывания, так называемое условие нерастяжимости твёрдого тела, скольжение без отрыва и т. п.

3. Кинодинамические: принуждают траектории соответствовать не только кинематическим ограничениям, но и уравнениям выбранной модели динамики системы (из-за того, что мы больше не можем пренебрегать инерцией объекта, добавляются условия на ускорение). Яркими представителями являются второй закон Ньютона и уравнение динамики вращательного движения.

По сложности среды задача планирования классифицируется следующим образом.

1. Имеются только статические препятствия.
   Обычно форма и расположение известны заранее и нам достаточно один раз их промоделировать, а дальше сконцентрироваться исключительно на построении траектории.

2. Кроме статических ещё возможны динамические препятствия.
   Форма и расположение таких препятствий считываются с датчиков системы и постоянно обновляются. Для успешного маневрирования в условиях меняющейся среды агенту помимо планирования собственного движения необходимо прогнозировать действия других участников и корректировать траекторию с целью избегания столкновений.

3. Многоагентная постановка.
   Предполагает координацию действий нескольких агентов для достижения общей цели, в ходе выполнения которой для получения оптимальных результатов агентам требуется постоянно согласовывать планы и синхронизировать свою работу.

Относительно используемого принципа существуют десятки различных подходов к задаче планирования. Мною были проанализированы три наиболее зарекомендовавших себя [1]:

1. Поиск по предварительно построенной дорожной карте (PRM).
   Заключается в дискретизации окружающего пространства графом с применением в дальнейшем алгоритмов поиска наилучшего пути в нём. Данный метод эффективен в средах с заранее известными препятствиями, но в случаях, когда агенту необходимо объехать преграду, которая не учитывалась при проектировании дорожной карты или проехать по участку, не охваченному графом дорожной сети, возникают трудности.

2. Инкрементальные методы на основе сэмплирования (RRT, RRG).
   Алгоритмы, основанные на этом принципе, постепенно строят дерево по случайным выборкам из пространства поиска, тем самым последовательно заполняя его. При этом на каждой итерации предпринимается попытка соединить имеющееся дерево с конечным множеством. От предыдущего метода его отличает отсутствие разделения на дискретизацию и поиск, и то и другое происходит на каждой итерации. Данный подход характеризуется высокой скоростью работы и приспосабливанием к изменяющейся обстановке.

3. Оптимизационные алгоритмы (Potential Fields).
   Обычно для методов такого типа изначально задан глобальный план (к примеру функция распределения потенциала в пространстве или ломанная, соединяющая начальную и конечную точки и т. п.) и задача состоит в том, чтобы построить гладкий путь, оптимизируя некоторый функционал по исходной концепции. Основными минусами являются необходимая предобработка и сходимость к локальным минимумам, выигрыш же в непрерывности получаемых управлений, качестве решений.

Представленная работа состоит в исследовании применения инкрементального алгоритма на основе сэмплирования RRT^* к задаче кинодинамического планирования со стационарной системой линейных дифференциальных ограничений и средой со статическими препятствиями. В данной статье предпринимается попытка уменьшить число "холостых" итераций алгоритма (тех, которые не приводят к образованию нового ребра в графе) путём использования внешних эллипсоидальных оценок множеств достижимости. Идея состоит в том, чтобы генерировать новые состояния в окрестности быстрой достижимости от имеющихся, что должно позволить нам повысить скорость расширения графа и снизить вычислительные затраты.

2 Постановка задачи

Пусть мы имеем ряд кинодинамических ограничений, представляющих собой линейную стационарную систему дифференциальных уравнений

$$\begin{cases} \dot{x}(t) = Ax(t) + Bu(t) + C, \\ x(t_0) = x_{\text{init}}, \\ x(t_1) \in X_{\text{goal}}. \end{cases} \tag{1}$$

Где $x(t) \in X$ — состояние системы в момент времени t, $u(t) \in \mathcal{U}$ — управляющее воздействие, $A \in \mathbb{R}^{n \times n}$, $B \in \mathbb{R}^{n \times m}$, $C \in \mathbb{R}^n$ — заданные постоянные матрицы, t_0 — начальный момент времени, t_1 — конечный момент времени (не фиксирован). Кроме того, нам заданы начальная точка x_{init}, целевое множество значений вектора состояний X_{goal}, множество допустимых управлений $\mathcal{U}$. За X_{obs} обозначим открытое подмножество X, определяющее область препятствий. X_{free} же, задающее доступную для манёвров область, примем равной $X \setminus X_{\text{obs}}$.

Задача состоит в том, чтобы при заданной области X, известном пространстве препятствий X_{obs}, начальном положении x_{init} и обозначенной целевой области X_{goal} найти программное управление $u(\cdot) \in \mathcal{U}$, которое минимизирует функционал

$$\mathcal{J}(u(\cdot)) = \int_{t_0}^{t_1} \langle x(t), Qx(t) \rangle + \langle u(t), Ru(t) \rangle \, \mathrm{d}t, \tag{2}$$

на беспрепятственных траекториях системы (1). Беспрепятственными будем называть решения $x(t) \in X_{\text{free}}, \forall t \in [t_0, t_1]$. В рассматриваемом функционале Q — положительно полуопределённая матрица, а R — положительно определённая матрица.

3 Алгоритм KRRT*

3.1 Базовый вариант

Algorithm 1: KRRT*

$V \leftarrow \{x_{init}\}$; $E \leftarrow \emptyset$;
for $i = 1, \ldots, n$ **do**
  $x_{rand} \leftarrow \texttt{SampleFree}$;
  $x_{nearest} \leftarrow \texttt{Nearest}(G = (V, E), x_{rand})$;
  $x_{new} \leftarrow \texttt{Steer}(x_{nearest}, x_{rand})$;
  if *not pathCollisionTest*$(x_{nearest}, x_{new})$ **then**
    $X_{near} \leftarrow \texttt{Near}(G = (V, E)), x_{new}, \min\{\gamma_{RRT^*}(\log(card(V))/card(V))^{\frac{1}{d}}, \eta\}$;
    $V \leftarrow V \cup \{x_{new}\}$;
    $x_{min} \leftarrow x_{nearest}$; $c_{min} \leftarrow Cost(x_{nearest}) + c(x_{nearest}, x_{new})$;
    foreach $x_{near} \in X_{near}$ **do**
      if *not pathCollisionTest*(x_{near}, x_{new}) *and* $Cost(x_{near}) + c(x_{near}, x_{new}) < c_{min}$ **then**
        $x_{min} \leftarrow x_{near}$; $c_{min} \leftarrow Cost(x_{near}) + c(x_{near}, x_{new})$;
      end
    end
    $E \leftarrow E \cup \{(x_{min}, x_{new})\}$;
    foreach $x_{near} \in X_{near}$ **do**
      if *not pathCollisionTest*(x_{new}, x_{near}) *and* $Cost(x_{new}) + c(x_{new}, x_{near}) < Cost(x_{near})$ **then**
        $x_{parent} \leftarrow \texttt{Parent}(x_{near})$;
        $E \leftarrow (E \setminus \{(x_{parent}, x_{near})\}) \cup \{(x_{new}, x_{near})\}$;
      end
    end
  end
end

Прежде чем перейти к подробному обсуждению алгоритма о котором можно почитать в [2], обозначим базовые функции:

- `SampleFree` — генератор равномерного распределения в X_{free}.
- $\texttt{Nearest}(G = (V, E), z)$ ищет ближайшую вершину графа к данной точке, то есть

$$\texttt{Nearest}(G, z) = \text{argmin}_{v \in V} dist(v, z).$$

- $\texttt{Steer}(z_1, z_2)$ строит оптимальную с точки зрения (2) траекторию, удовлетворяющую заданной системе дифференциальных ограничений, при этом стартующую в z_1 и заканчивающуюся в некоторой окрестности z_2.
- $\texttt{pathCollisionTest}(z_1, z_2)$ — функция, проверяющая траекторию (z_1, z_2) на наличие столкновений, то есть отвечающая на вопрос: лежит ли $x(t)$ в множестве X_{free} для всех $t \in [t_0, T]$.

- $\texttt{Near}(G, z, \eta)$ возвращает множество соседних точек, отдалённых от рассматриваемой не более чем на η.

$$\texttt{Near}(G = (V, E), z, \eta) = \{v \in V \mid v \in B_{z,\eta}\}.$$

- $Cost(z)$ — стоимость оптимальной траектории, ведущей от корневой вершины к данной.

3.2 Общая схема работы алгоритма и его оптимизация

Глобальная схема работы $KRRT^*$, выглядит так: инициализированный единственной вершиной x_{init} алгоритм итеративно строит дерево беспрепятственных траекторий, сначала выбирая состояния из допустимого конфигурационного пространства, а затем соединяя их с деревом. Останавливается по истечению числа итераций. Про достаточные условия асимптотической оптимальности для алгоритмов рассматриваемого типа можно прочитать в [3].

Теперь чуть более детально разберём k-ю итерацию.

1. Генерируется точка x_{rand} равномерно в пространстве X_{free}.
2. Ищется ближайшая к ней из уже имеющихся в графе вершин.
3. Строится оптимальная траектория от ближайшей к сгенерированной (не обязана достигать желаемого конечного состояния), конец сохраняем в x_{new}.
4. Если траектория задевает препятствие, то выполняется переход на следующую итерацию цикла, иначе определяется множество X_{near} — ближайших вершин, отстоящих от x_{new} не более чем на определённое расстояние.
5. Соединяем x_{new} с одной вершиной из множества X_{near}, через которую проходит минимальный допустимый путь от корня к x_{new}, добавляем это ребро в граф.
6. Далее на множестве X_{near} выполняем операцию перемонтирования: если от корня через x_{new} в какую-то из вершин x_{near} можно беспрепятственно добраться за меньшую стоимость, чем $Cost(x_{near})$, то ребро соединяющее x_{near} со своим родителем удаляется из графа, а (x_{new}, x_{near}) в граф добавляется.

Суть предлагаемой мною эвристики заключается в добавлении к алгоритму перед генерацией состояния ещё одной функции `EllipsoidalEstimate`, которая строит внешнюю эллипсоидальную оценку множества достижимости из вершины графа за время T_{max}[1]. Затем происходит сэмплирование равномерно не на всём пространстве, а только в эллипсоиде, что

[1]Эллипсоидальные оценки подразумевают эллипсоидальное ограничение на допустимые управления. Если брать ограничения, зависящие от времени, вида $u \in \mathcal{E}(m(t), M(t))$, то операция перемонтирования потеряет свой смысл (детей пересоединенной вершины придётся удалять, ведь время прибытия в их предка изменится, а значит неизбежен пересчет траекторий). Поэтому мною рассматриваются ограничения вида $u \in \mathcal{E}(m, M)$. В таком случае матрица эллипсоида будет совпадать у всех вершин, различным будет лишь вектор сдвига.

сильно повышает шансы на достижение нового состояния, при этом даже в случае неудачи на построение будет потрачено небольшое (параметр T_{max}) время. Для базового же варианта, не подстраивающегося под особенности системы дифференциальных ограничений, вероятность сгенерировать удалённую точку, до которой мы не сможем добраться или же будем добираться крайне долго — выше.

4 Аналитический анализ основных функций

4.1 Построение внешних эллипсоидальных оценок множества достижимости

4.1.1 Внешняя эллипсоидальная оценка для суммы эллипсоидов

Сначала научимся аппроксимировать сумму m эллипсоидов. Покажем, что для

$$\mathcal{E}_+(0, Q_+) \supseteq \sum_{k=1}^{m} \mathcal{E}(0, Q_k)$$

его матрицу Q_+, зависящую от вектора $p[m]$ с m компонентами, можно задать как

$$Q_+(p[m]) = \left(\sum_{k=1}^{m} p_k\right)\left(\sum_{k=1}^{m} p_k^{-1} Q_k\right).$$

Сделаем это через опорные функции соответствующих множеств

$$\rho^2(l \mid \mathcal{E}(0, Q_+)) = \langle l, Q_+ l\rangle = \sum_{k=1}^{m} \langle l, Q_k l\rangle + \sum_{i=1}^{m} \sum_{j=1, j\neq i}^{m} \frac{p_j \langle l, Q_i l\rangle}{p_i} \geq$$

$$\geq \{\text{среднее арифметическое} \geq \text{среднего геометрического}\} \geq$$

$$\geq \sum_{k=1}^{m} \langle l, Q_k l\rangle + 2\sum_{i=1}^{m-1} \sum_{j=i+1}^{m} \sqrt{\langle l, Q_i l\rangle \langle l, Q_j l\rangle} = \rho^2\left(l \;\middle|\; \sum_{k=1}^{m} \mathcal{E}(0, Q_k)\right).$$

Заметим, что равенство по направлению l достигается при

$$p_k = \sqrt{\langle l, Q_k l\rangle}, \quad k = \overline{1, m}.$$

4.1.2 Внешняя эллипсоидальная оценка для интеграла по эллипсоиду

Перейдём теперь к оценке интеграла с непрерывными $q(t)$ и $Q(t)$:

$$I = \int_{t_0}^{t} \mathcal{E}(q(\tau), Q(\tau))\, \mathrm{d}\tau.$$

Так как эллипсоид $\mathcal{E}(q, Q)$ — непрерывное многозначное отображение по (q, Q), а кроме того является замкнутым и ограниченным множеством в $\mathbb{R}^n$ (то есть компактно), то мы можем его проинтегрировать на отрезке. Для аппроксимации интеграла будем использовать интегральные суммы на равномерных сетках с диаметром разбиения δ:

$$I(N) = \delta \cdot \sum_{k=1}^{N} \mathcal{E}(q(\tau_k), Q(\tau_k)),$$

тогда

$$q_+(N) = \delta \cdot \sum_{k=1}^{N} q(\tau_k),$$

$$Q_+(N) = \left(\delta \cdot \sum_{k=1}^{N} p(\tau_k)\right)\left(\delta \cdot \sum_{k=1}^{N} p^{-1}(\tau_k) Q(\tau_k)\right)$$

соответствующие формулы для внешней оценки. Если устремить $N \to +\infty$, то в силу непрерывности имеет место сходимость аппроксимирующих эллипсоидов

$$q_+(N) \to q_+, \ Q_+(N) \to Q_+.$$

Причём в силу непрерывности опорной функции по множеству, предельный эллипсоид будет внешней оценкой для интеграла. Получим, что

$$q_+(t) = \int_{t_0}^{t} q(\tau)\,\mathrm{d}\tau,$$

$$Q_+(t) = \left(\int_{t_0}^{t} p(\tau)\,\mathrm{d}\tau\right)\left(\int_{t_0}^{t} p^{-1}(\tau) Q(\tau)\,\mathrm{d}\tau\right).$$

При этом опять же, равенство опорных функций по направлению l достигается тогда и только тогда, когда

$$p(\tau) = \sqrt{\langle l, Q(\tau) l\rangle} \quad \tau \in [t_0, t].$$

4.1.3 Внешняя эллипсоидальная оценка для множества достижимости

Построим внешнюю эллипсоидальную оценку ко множеству достижимости $X[t]$ системы

$$\begin{cases} \dot{x}(t) = Ax(t) + Bu(t) + C, \\ x(t_0) = x_0, \\ u(t) \in \mathcal{E}(q(t), P(t)). \end{cases}$$

Множество достижимости может быть представлено в виде:

$$X[t] = X(t, t_0)x_0 + \int_{t_0}^{t} X(t, \tau)(B\mathcal{E}(q(\tau), P(\tau)) + C)\,\mathrm{d}\tau =$$

$$= \left[X(t,t_0)x_0 + \int_{t_0}^{t} X(t,\tau)C\,\mathrm{d}\tau\right] + \int_{t_0}^{t} X(t,\tau)B\mathcal{E}(q(\tau),P(\tau))\,\mathrm{d}\tau =$$

$$= v(t) + \int_{t_0}^{t} \mathcal{E}(X(t,\tau)Bq(\tau), X(t,\tau)BP(\tau)B^TX^T(t,\tau))\,\mathrm{d}\tau.$$

Тогда, исходя из проделанных ранее выкладок, внешний аппроксимирующий эллипсоид будет иметь вид

$$q_+(t) = v(t) + \int_{t_0}^{t} X(t,\tau)Bq(\tau)\,\mathrm{d}\tau, \tag{3}$$

$$X_+[t] = \left(\int_{t_0}^{t} p(\tau)\,\mathrm{d}\tau\right)\left(\int_{t_0}^{t} p^{-1}(\tau)X(t,\tau)BP(\tau)B^TX^T(t,\tau)\,\mathrm{d}\tau\right). \tag{4}$$

И кроме того,

$$p_t(\tau) = \left\langle l(t), X(t,\tau)BP(\tau)B^TX^T(t,\tau)l(t)\right\rangle^{\frac{1}{2}}.$$

Индекс t — зависимость от момента сечения пучка всевозможных траекторий, для которого строим оценку. То есть мы не можем для фиксированного l сделать хорошей оценки, так как для каждого t придётся делать перерасчёт, что влечёт большую вычислительную сложность. В виду чего предлагается определить зависимость $l(t)$:

$$l^*(t) = X(t_0,t)^Tl_0,$$

тогда, подставив её в предыдущее равенство, получим:

$$p(\tau) = \left\langle l_0, X(t_0,\tau)BP(\tau)B^TX^T(t_0,\tau)l_0\right\rangle^{\frac{1}{2}}.$$

Также, для эффективности вычислений, если нам необходимо иметь оценку множества достижимости в каждый момент времени от начального до некоторого t, то удобней будет не пересчитывать интегралы (3), (4), а решить соответствующие дифференциальные уравнения. Выведем их. Обозначим

$$M_1(t) = \int_{t_0}^{t} p(\tau)\,\mathrm{d}\tau,$$

$$M_2(t) = \int_{t_0}^{t} p^{-1}(\tau)X(t,\tau)BP(\tau)B^TX^T(t,\tau)\,\mathrm{d}\tau.$$

Тогда $X_+[t] = M_1(t)M_2(t)$. Получаем систему

$$\begin{cases} \dot{M}_1(t) = p(t), \\ \dot{M}_2(t) = \dfrac{BP(t)B^T}{p(t)} + AM_2(t) + M_2(t)A^T, \\ \dot{X}_+(t) = p(t)M_2(t) + AX_+(t) + X_+(t)A^T + M_1(t)\dfrac{BP(t)B^T}{p(t)}, \\ M_1(t_0) = 0, \\ M_2(t_0) = O, \\ X_+(t_0) = O. \end{cases}$$

$$\begin{cases} \dot{q}_+(t) = Aq_+(t) + Bq(t) + C, \\ q_+(t_0) = x_0. \end{cases}$$

4.2 Генерация равномерного распределения в гиперэллипсоиде (SampleFree)

4.2.1 Эквивалентность задач генерации равномерного распределения внутри единичной гиперсферы и гиперэллипсоида

После построения внешней эллипсоидальной оценки, нам бы хотелось научиться равновероятно выбирать точки из неё. Покажем, что задачу о равномерном сэмплировании в гиперэллипсоиде можно свести к таковой для гиперсферы. Начнём с того, что представим гиперэллипсоид $\mathcal{E}(q, Q)$ линейным преобразованием над единичной гиперсферой $S(0, 1)$.

$$\rho(l \mid \mathcal{E}(q, Q)) = \rho\left(l \;\middle|\; q + Q^{\frac{1}{2}}\mathcal{E}(0, I)\right) = \rho\left(l \;\middle|\; q + Q^{\frac{1}{2}}S(0, 1)\right).$$

Следовательно, в силу однозначного соответствия между опорными функциями и выпуклыми, замкнутыми множествами, получаем

$$\mathcal{E}(q, Q) = q + Q^{\frac{1}{2}}S(0, 1). \tag{5}$$

Плотность равномерного распределения в n-мерной единичной сфере

$$p_{ball}(x) = \begin{cases} \dfrac{1}{\zeta_n}, & \forall x \in X_{ball}, \\ 0, \text{иначе}. \end{cases}$$

Исходя из (5)

$$x_{ellipse} = g(x_{ball}) = Q^{\frac{1}{2}}x_{ball} + q,$$

несложно найти плотность такой случайной величины через Якобиан перехода

$$p_{ellipse}(x) = p_{ball}(g^{-1}(x)) \left| \det \left\{ \frac{\mathrm{d}g^{-1}(y)}{\mathrm{d}y} \Big|_{y=x} \right\} \right|.$$

Следовательно,

$$p_{ellipse}(x) = \begin{cases} \dfrac{1}{\zeta_n} \cdot \left|Q^{-\frac{1}{2}}\right|, & \forall x \in X_{ellipse}, \\ 0, \text{иначе.} \end{cases}$$

Для положительно определённой матрицы квадратный корень существует и единственен, кроме того, его собственные значения являются корнями из соответствующих исходного оператора. При этом любую квадратичную форму можно привести к диагональному виду, а значит, учитывая подобие любых двух представлений оператора в различных базисах определитель $Q^{\frac{1}{2}}$ сохраняется и равен корню из произведения собственных значений Q. Таким образом $\frac{1}{\zeta_n} \cdot |Q^{-\frac{1}{2}}|$ — обьём гиперэллипсоида (тут учли ещё что у обратной матрицы обратный спектр и что собственные значения матрицы эллипсоида являются квадратами его полуосей). Таким образом мы показали, что нам достаточно лишь научиться равновероятно генерировать точки в единичной гиперсфере.

Заметим, что схожие рассуждения можно провести и для разложения Холецкого $Q = LL^T$, получив плотность

$$p_{ellipse}(x) = \begin{cases} \dfrac{1}{\zeta_n} \cdot \left|L^{-1}\right|, & \forall x \in X_{ellipse}, \\ 0, \text{иначе.} \end{cases}$$

Единственное, вычисления будут происходить на больших размерностях сильно быстрее, что является несомненным преимуществом такого подхода.

4.2.2 Генерация равномерного распределения в единичной гиперсфере

Для этого воспользуемся методом Мюллера. Будем равномерно выбирать сначала направление в пространстве, а затем радиус. Сэмплирование направления основано на следующем факте: для случайного вектора с независимыми, стандартно нормально распределёнными компонентами функция плотности имеет вид

$$f(x_1, \ldots, x_n) = \frac{1}{(2\pi)^{\frac{n}{2}}} \cdot e^{-\frac{1}{2}||x||^2},$$

то есть плотность не зависит от направления, а только лишь от длины вектора и является одинаковой на одном расстоянии. Радиус же задаётся как

$$r^{\frac{1}{d}}, \quad r \sim U[0, 1],$$

где d — размерность пространства [4].

4.3 Функция рулевого управления (Steer)

Затронем тему реализации функции рулевого управления $\mathrm{Steer}(\mathrm{q}_1, \mathrm{q}_2)$, назначение которой в построении оптимального сегмента пути, удовлетворяющего заданным дифференциальным ограничениям, от состояния q_1 к состоянию q_2. При этом отметим, что построенный путь не обязан достигать q_2.

Перейдём к формальной постановке задачи. Имеется система дифференциальных ограничений

$$\begin{cases} \dot{x}(t) = Ax(t) + Bu(t) + C, \\ x(\tau) = q_1. \end{cases} \tag{6}$$

А также задано желаемое положение в конечный момент времени $x(T) = q_2$, T — неизвестно. Необходимо определить программное управление $u(\cdot) \in \mathcal{U}$, которое минимизирует функционал (2) на траекториях системы (6) и ведёт траекторию к q_2. Чтобы учесть желаемое конечное состояние рассмотрим ту же задачу при дифференциальных ограничениях (6), но с функционалом

$$\mathcal{K}(u(\cdot)) = \int_{\tau}^{T} \langle x(t), Qx(t) \rangle + \langle u(t), Ru(t) \rangle \, \mathrm{d}t + \langle x(T) - q_2, S(x(T) - q_2) \rangle, \tag{7}$$

где терминальный член $\langle x(T) - q_2, S(x(T) - q_2) \rangle$ имеет матрицу $S > 0$.

4.3.1 Построение решения для фиксированного конечного времени

Решим поставленную задачу, используя метод динамического программирования. Осуществим поиск оптимального управления через функцию Беллмана. По определению она задаётся как

$$V(t, x) = \min_{u(\cdot)} \{\mathcal{K}(t, x, u(\cdot))\}. \tag{8}$$

В то же время функция цены нашей задачи должна удовлетворять попятному уравнению Беллмана:

$$\begin{cases} \frac{\partial V}{\partial t} + \inf\limits_{u(\cdot)} \left\{ \left\langle \frac{\partial V}{\partial x}, Ax + Bu + C \right\rangle + \left\langle x, Qx \right\rangle + \left\langle u, Ru \right\rangle \right\} = 0, \\ V(T, x) = \left\langle x - q_2, S(x - q_2) \right\rangle. \end{cases} \tag{9}$$

Покажем, что для $f(u) = \left\langle \frac{\partial V}{\partial x}, Ax + Bu + C \right\rangle + \left\langle x, Qx \right\rangle + \left\langle u, Ru \right\rangle$ инфимум достигается при произвольной непрерывно дифференцируемой $V(t, x)$, это будет означать по теореме о верификации, что любая гладкая функция, являющаяся решением (9) удовлетворяет (8). Выпишем

необходимое условие экстремума $f(u)$:

$$\nabla f = 0 \Leftrightarrow B^T \frac{\partial V}{\partial x} + 2Ru = 0 \Rightarrow$$

$$\Rightarrow u^*(t,x) = -\frac{1}{2} R^{-1} B^T \frac{\partial V}{\partial x}. \tag{10}$$

Теперь достаточное условие минимума:

$$d^2 f(u^*) > 0 \Leftrightarrow R > 0,$$

что выполнено из условия. Поэтому нам достаточно найти хотя бы какое-то гладкое решение (9), а после по нему из (10) восстановить u^*[2].

Функцию цены будем искать в виде

$$V(t,x) = \langle x - y(t), P(t)(x - y(t)) \rangle + k(t),$$

где $y(t), k(t)$ — гладкие функции, зависящие от времени, $P(t)$ — непрерывно дифференцируемая матричная функция, в каждый момент времени представляющая собой положительно определённую квадратичную форму. Тогда

$$\frac{\partial V}{\partial t} = \langle x - y, \dot{P}(x - y) \rangle - 2\langle \dot{y}, P(x - y) \rangle + \dot{k}, \tag{11}$$

$$\frac{\partial V}{\partial x} = 2P(x - y). \tag{12}$$

Подставим выражения (11) и (12), а также значение оптимального управления (10) в уравнение Беллмана (9), получим:

$$\langle x - y, \dot{P}(x - y) \rangle - 2\langle \dot{y}, P(x - y) \rangle + \dot{k} + \big\langle 2P(x - y), Ax + C \big\rangle - \\ - \big\langle P(x - y), BR^{-1}B^T P(x - y) \big\rangle + \big\langle x, Qx \big\rangle = 0. \tag{13}$$

Отметим следующие удобные представления:

$$2\big\langle P(x - y), Ax + C \big\rangle = \big\langle PA(x - y), x - y \big\rangle + \big\langle A^T P(x - y), x - y \big\rangle + 2\big\langle P(x - y), Ay + C \big\rangle,$$

$$\big\langle x, Qx \big\rangle = \big\langle x - y, Q(x - y) \big\rangle + 2\big\langle y, Q(x - y) \big\rangle + \big\langle y, Qy \big\rangle.$$

[2]На практике столкнёмся с ограничением на $u \in \mathcal{E}(p, P)$, тогда придётся проверять принадлежность полученного u^* множеству $\mathcal{U}$

Используя их перепишем (13) в виде

$$\left\langle x-y, \left[\dot{P}+PA+A^TP-PBR^{-1}B^TP+Q\right](x-y)\right\rangle + \\ + \left\langle x-y, [-2P\dot{y}+2PAy+2PC+2Qy]\right\rangle + \dot{k} + \langle y, Qy\rangle = 0,$$

откуда перейдём к трём системам:

$$\begin{cases} \dot{P}+PA+A^TP-PBR^{-1}B^TP+Q=0, \\ P(T)=S. \end{cases} \tag{14}$$

$$\begin{cases} \dot{y}-Ay-C-P^{-1}Qy=0, \\ y(T)=q_2. \end{cases} \tag{15}$$

$$\begin{cases} \dot{k}+\langle y, Qy\rangle = 0, \\ k(T)=0. \end{cases} \tag{16}$$

Заметим, что граничные условия представленных систем соответствуют второму уравнению в (9), записывающемуся как:

$$\langle x(T)-y(T), P(T)(x(T)-y(T))\rangle + k(T) = \big\langle x(T)-q_2, S(x(T)-q_2)\big\rangle.$$

А значит функция, полученная из решений систем (14)-(16), удовлетворяет попятному уравнению Беллмана и, следовательно, определению (8). Тогда оптимальное управление

$$u^*(t,x) = -R^{-1}B^TP(t)(x-y(t)), \tag{17}$$

является непрерывной функцией по (t,x) и линейной по x. Подставив найденное выражение в (6), мы получим систему дифференциальных уравнений с непрерывной по (t,x) и линейной по x правой частью, в силу чего будут выполнены условие липшица по x на всём $\mathbb{R}^n$, условие сублинейного роста. Поэтому на $[\tau,T]$ найдётся гладкая оптимальная траектория и притом только одна. Зная оптимальную траекторию можно выяснить искомое программное управление, подставив решение в (17).

Проговорим необходимые ограничения на исходную систему, которые гарантировали бы нам "хорошие" решения (14)-(16). Линейные системы (15), (16) можно представить в виде $\dot{x} = A(t)x + C(t)$ с известными конечными условиями, где $A(t)$, $C(t)$ — непрерывные матричные функции, следовательно, аналогично тому, как было сделано в абзаце выше, можно показать, что они имеют гладкие решения на всём рассматриваемом $[\tau,T]$ и притом единственные. С (14) всё немного иначе. Матричное уравнение Риккати имеет решение и к тому же единственное тогда и только тогда, когда выполнены следующие условия [5]:

1. Q, S — положительно полуопределены.

2. R — положительно определена.

3. (A, B) — стабилизируема.

4. $(A, Q^{\frac{1}{2}})$ — детектируема.

Требования 1-2 реализованы из постановки задачи, условия 3-4 для простоты проверки удобно заменить на более сильные: управляемость (A, B) и наблюдаемость $(A, Q^{\frac{1}{2}})$.

Подытожив всё вышесказанное, подчеркнём, что оптимальный сегмент пути от q_1 к q_2, удовлетворяющий заданным дифференциальным ограничениям будет построен нашей функцией, если $Q \geqslant 0$, $S \geqslant 0$, $R > 0$, (A, B) — управляема и $(A, Q^{\frac{1}{2}})$ — наблюдаема. Поэтому в начале функции реализуем проверку корректности введённых данных:

- Определённость матриц: isequal(R, R′) && all(eig(R) > 0), аналогично для Q, S.
- Управляемость: rank(ctrb(A, B)) == n.
- Наблюдаемость: rank(obsv(A, $Q^{\frac{1}{2}}$)) == n.

Стоит сказать, что тут мы воспользовались критериями управляемости и наблюдаемости для линейных стационарных систем, а также критерием знакоопределённости квадратичной формы через собственные значения.

4.3.2 Построение решения для свободного конечного времени

Для оптимальной пары (x^*, u^*), являющейся решением поставленной в этом разделе проблемы справедлив принцип максимума Понтрягина. Обозначим за $\psi(t)$ — сопряжённую вектор-функцию, $\mathcal{H}(\psi, x, u)$ — функцию Гамильтона-Понтрягина рассматриваемой задачи, $\phi(T, x(T))$ — терминальный член функционала (7), тогда

$$\mathcal{H}(\psi, x, u) = \langle x, Qx\rangle + \langle u, Ru\rangle + \langle \psi, Ax + Bu + C\rangle, \tag{18}$$

$$\dot{\psi}(t) = -\frac{\partial \mathcal{H}(\psi, x, u)}{\partial x}\bigg|_{(x=x^*, u=u^*)} = -A^T\psi(t) + 2Qx^*(t), \tag{19}$$

$$\frac{\mathrm{d}\mathcal{H}(\psi(t), x(t), u(t))}{\mathrm{d}t}\bigg|_{(\psi=\psi(t), x=x^*(t), u=u^*(t))} = \left\langle \psi, \frac{\partial f(t, x^*, u^*)}{\partial t}\right\rangle = 0 \;\Rightarrow\; \mathcal{H}(t) \equiv \text{const}, \tag{20}$$

$$\mathcal{H}(T) = 0. \tag{21}$$

Отметим, что выражение (20) выполнено потому что $\frac{\partial f(t,x,u)}{\partial t} = 0$ является непрерывной, а условие (21) на правом конце для функции $\mathcal{H}(t)$ справедливо, так как T свободно и $\frac{\partial \phi(T,x)}{\partial T} = 0$.

Пользуясь (20) и (21) получим $\mathcal{H}(t) \equiv 0\ \forall t \in [\tau, T]$. Из связи уравнения Беллмана и принципа максимума Понтрягина $\psi(t) = \frac{\partial V(t,x(t))}{\partial x}$, учитывая это и (9), (18), получим необходимое

условие на функцию Гамильтона-Понтрягина в начальный момент времени τ:

$$0 = \mathcal{H}(\tau) = -\frac{\partial V(t, x(\tau))}{\partial t}\bigg|_{t=\tau} = \{(11)\} =$$

$$= \langle x(\tau) - y(\tau), \dot{P}(\tau)(x(\tau) - y(\tau))\rangle - 2\langle \dot{y}(\tau), P(\tau)(x(\tau) - y(\tau))\rangle + \dot{k}(\tau) =$$

$$= \{(14), (15), (16)\} = \left\langle y(\tau), Qy(\tau)\right\rangle + 2\langle \left(A + P^{-1}(\tau)Q\right) y(\tau) + C, P(\tau)(x(\tau) - y(\tau))\rangle -$$

$$- \left\langle x(\tau) - y(\tau), \left[-P(\tau)A - A^T P(\tau) + P(\tau)BR^{-1}B^T P(\tau) - Q\right](x(\tau) - y(\tau))\right\rangle. \qquad (22)$$

Используем его для нахождения искомого конечного момента. Решая в обратном времени системы (14), (15), будем сохранять времена обнулений (22), получим набор временных интервалов вида (τ'_k, T), где τ'_k — время наступления k-го события. В силу эквивалентности своих решений для систем (14) и (15) на $(\tau, \tau + (T - \tau'_k))$ и (τ'_k, T) (начальные условия сдвигаются соответственно) нам достаточно лишь сравнить решения (x_k, u_k) соответствующие $(\tau, \tau + (T - \tau'_k))$ и определить среди них оптимальное.

4.4 Поиск столкновений (pathCollisionTest)

4.4.1 Проверка отдельного состояния

В процессе своего движения робот должен уметь объезжать встречающиеся у него на пути преграды. Будем предполагать, что препятствия представляют собой выпуклые многоугольники по первым двум координатам (в $\mathbb{R}^2$) и полностью заполняют оставшиеся компоненты фазового вектора, являются неподвижными. Тогда доступная для манёвров область является невыпуклой и односвязной. Примером такой ситуации может служить объезд колесным роботом предприятия с целью диагностики имеющихся на нём механизмов.

Программная реализация выглядит следующим образом:

- В начале программы пользователь задаёт количество (N) преград и координаты их вершин.

- После происходит предобработка полученных данных. А именно: у каждого препятствия его узлы сортируются по полярному углу относительно соответствующей вершины O_i (асимптотика $O(n_i \log(n_i))$ по времени, $O(n_i)$ — память), имеющей минимальную координату по x_2 из точек с наименьшей координатой по x_1 (относительно нижней из самых левых у текущего многоугольника).

- Далее, когда нам нужно выяснить, принадлежит ли некоторая точка Q пространству X_{free}, запускается цикл по i от 1 до N, в котором

    1. Используя бинарный поиск по углу ($O(\log(n_i))$), определяем две соседние вершины L_i и R_i i-го многоугольника такие, что полярный угол рассматриваемой точки Q лежит между полярными углами L_i и R_i.

2. Узнаём, лежит ли Q внутри треугольника $O_iL_iR_i$ (образует ли R_i, L_i, Q поворот против часовой). Это можно сделать за $O(1)$ через ориентированную площадь треугольника:

$$2S_{\triangle} = \begin{vmatrix} L_{i,1} - R_{i,1} & L_{i,2} - R_{i,2} \\ Q_1 - R_{i,1} & Q_2 - R_{i,2} \end{vmatrix},$$

где значение большее, либо равное нуля, будет говорить о столкновении.

Таким образом на предварительную обработку разово тратится $O\left(\sum_{i=1}^{N} n_i \log(n_i)\right)$ по времени, $O\left(\sum_{i=1}^{N} n_i\right)$ по памяти, на запрос: $O\left(\sum_{i=1}^{N} \log(n_i)\right)$, $O(1)$ соответственно.

4.4.2 Проверка всего пути

Во избежание столкновений с препятствиями, сегменты пути, возвращаемые функцией `Steer`, обязаны лежать в X_{free}. Для этого реализуем функцию, проверяющую наличие преград вдоль траектории. Сделать это через имеющуюся `ObstacleFree` поточечно не получится, так как в таком случае потребуется континуальное число вызовов функции. Поэтому предлагается брать в рассмотрение не все точки, а их конечное число. При таком подходе нам бы хотелось быть уверенными в том, что на невыбранных участках столкновения также отсутствуют. Ввиду чего воспользуемся надёжным, зарекомендовавшим себя на практике, алгоритмом проверки пути на основе последовательности Ван Дер Корпута [6] с остановкой при попадании в препятствие.

5 Программная реализация

По схеме, указанной на пятой странице, реализуем три алгоритма: `RRT*` — версия с функционалом равным длине пути, не учитывающая дифференциальные ограничения, `KRRT*` — реализация с линейно-квадратичным функционалом, учитывающая кинодинамические условия и `MyKRRT*` — вариант `KRRT*` с предлагаемой эвристикой. Продемонстрируем примеры их работы, а также сравним полученные результаты. На всех рисунках ниже зелёным цветом обозначена целевая область X_{goal}, серым — препятствия X_{obs}, оранжевым — построенный граф, чёрным — оптимальная траектория, красным — начальное состояние.

5.1 Пример 1

Цель данного примера продемонстрировать способности алгоритмов к исследованию пространства. Пусть нам заданы следующие параметры системы

$$A = \begin{bmatrix} 10 & -10 \\ -25 & 15 \end{bmatrix}, B = \begin{bmatrix} 12 & -3 \\ -11 & 10 \end{bmatrix}, C = \begin{bmatrix} 0 \\ 0 \end{bmatrix}, R = Q = S = \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix}, x_0 = \begin{bmatrix} 1 \\ 1 \end{bmatrix}, t_0 = 0,$$

$T_{max} = 0.1, \texttt{count} = 10001, \eta = 2, m = \begin{bmatrix} 0 \\ 0 \end{bmatrix}, M = \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix}, \texttt{dimension} = 2,$ X задаётся как $\texttt{center} = \begin{bmatrix} 0 \\ 0 \end{bmatrix}, \texttt{side} = 1, X_{goal}, X_{obs}$ — квадраты с центрами в $\begin{bmatrix} 0.1 \\ 0.1 \end{bmatrix}, \begin{bmatrix} 0.5 \\ 0.5 \end{bmatrix}$, сторонами 0.2.

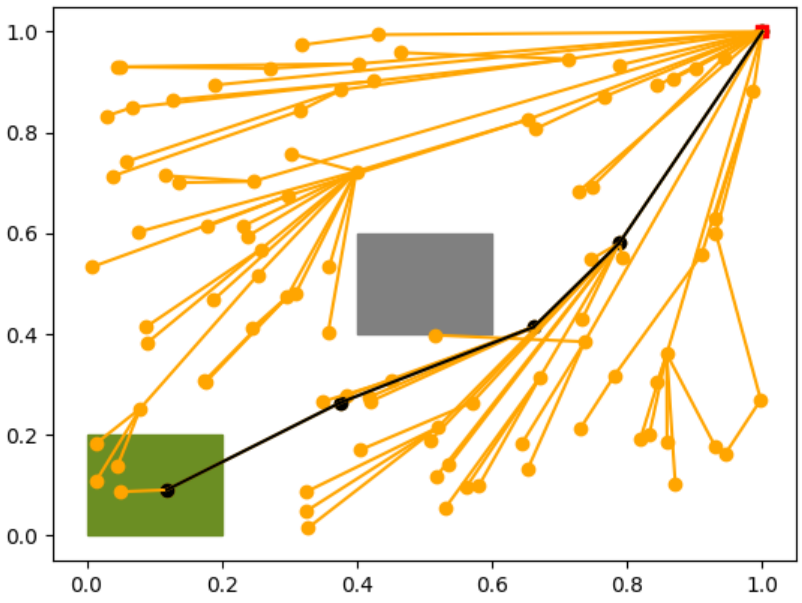

Рис. 1: RRT*, $N = 100$

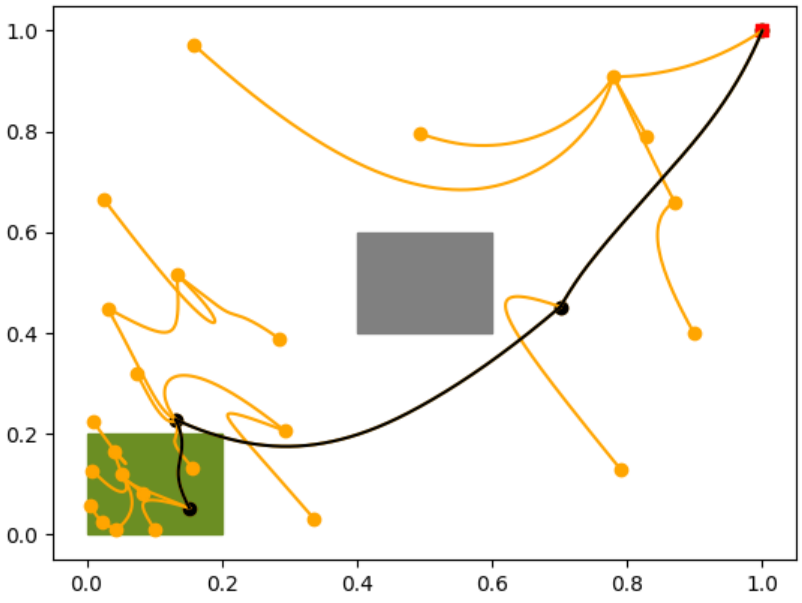

Рис. 2: KRRT*, $N = 1000$

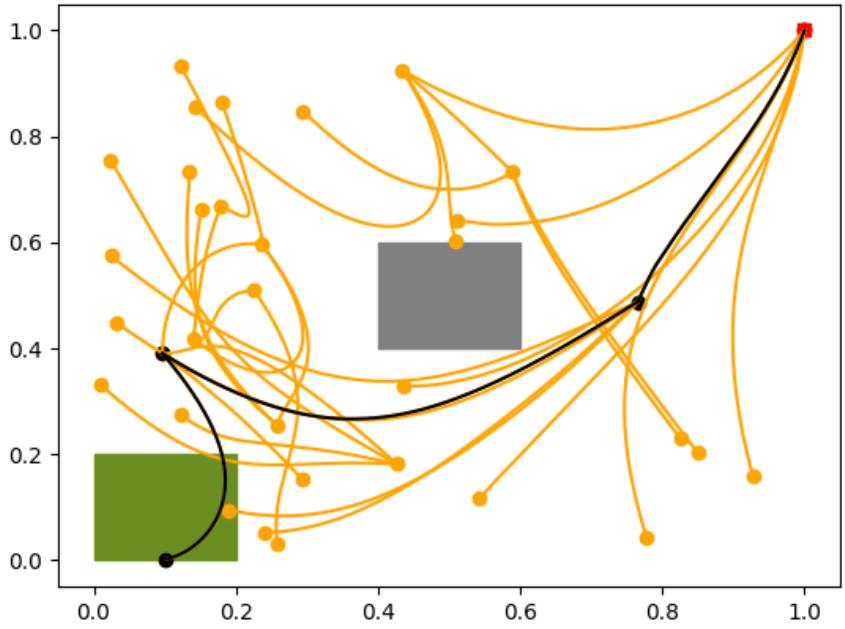

Рис. 3: MyKRRT*, $N = 1000$

	RRT*	KRRT*	MyKRRT*
Ср. число вершин (доля)	100 / 100	24 / 1000	35 / 1000
Ср. время работы	9.59s	1m 2s	1m 24s
Достижение X_{goal} (доля)	100/100	93 / 100	91 / 100

Таблица 1: Сравнение результатов алгоритмов на первом примере

Как видно из представленных графиков, у алгоритмов планирования с учётом дифференциальных ограничений способность присоединять к графу новые состояния сильно слабее. Чего и следовало ожидать, ведь из-за кинодинамических условий, наложенных на систему, часто мы не можем даже "сдвинуться в нужную сторону". Лучше всех справился алгоритм RRT*, правда простейший функционал (длина) и неприспособленность к дифференциальным ограничениям делают его непригодным для практического использования. Если сравнивать KRRT* и MyKRRT*, то второй алгоритм имеет небольшие преимущества над базовой реализацией, а именно: шире исследует пространство и соединяет большее число точек (что зависит от правильно подобранных параметров T_{max}, l — направление касания внешней оценки).

5.2 Пример 2

Теперь же оценим насколько хорошо алгоритмы умеют преодолевать узкие места.

$$A = \begin{bmatrix} 0 & -1 \\ -5 & 3 \end{bmatrix}, B = \begin{bmatrix} -7 & 6 \\ 0 & -5 \end{bmatrix}, C = \begin{bmatrix} 0 \\ 0 \end{bmatrix}, R = Q = S = \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix}, x_0 = \begin{bmatrix} 0.5 \\ 0.5 \end{bmatrix}, t_0 = 0,$$

$$T_{max} = 0.1, \texttt{count} = 10001, \eta = 2, m = \begin{bmatrix} 0 \\ 0 \end{bmatrix}, M = \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix}, \texttt{dimension} = 2, \texttt{center} = \begin{bmatrix} 0 \\ 0 \end{bmatrix}, \texttt{side} = 1,$$

$$X_{goal} \text{ — квадрат с центром в } \begin{bmatrix} 0.1 \\ 0.1 \end{bmatrix} \text{ и стороной } 0.2,$$

$$X_{obs} \text{ — пара прямоугольников с центрами в } \begin{bmatrix} 0.35 \\ 0.35 \end{bmatrix}, \begin{bmatrix} 0.15 \\ 0.35 \end{bmatrix}, \text{ и сторонами}$$

$0.5, 0.2$ (высота, ширина) и $0.1, 0.2$ соответственно.

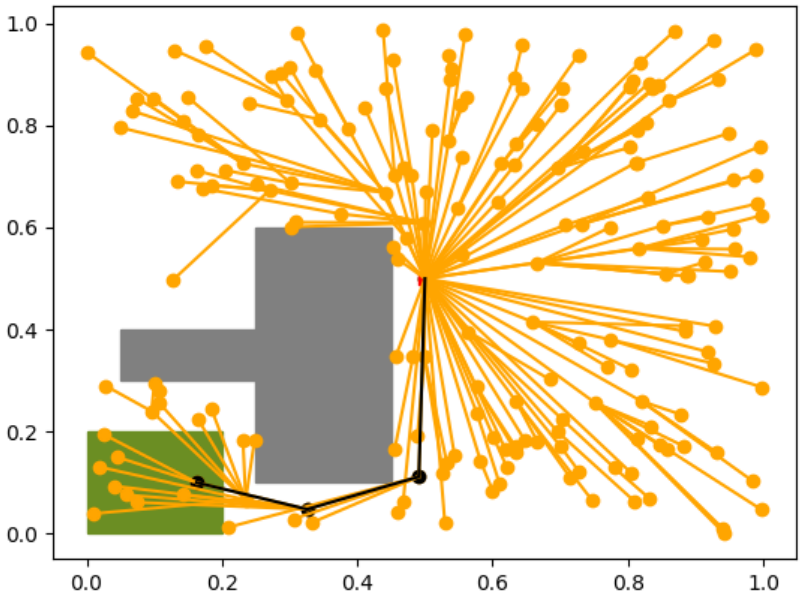

Рис. 4: RRT*, $N = 200$

	RRT*	KRRT*	MyKRRT*
Ср. число вершин (доля)	189 / 200	32 / 250	55 / 250
Ср. время работы	26.7s	1m 29s	2m 15s
Достижение X_{goal} (доля)	100/100	71 / 100	67 / 100

Таблица 2: Сравнение результатов алгоритмов на втором примере

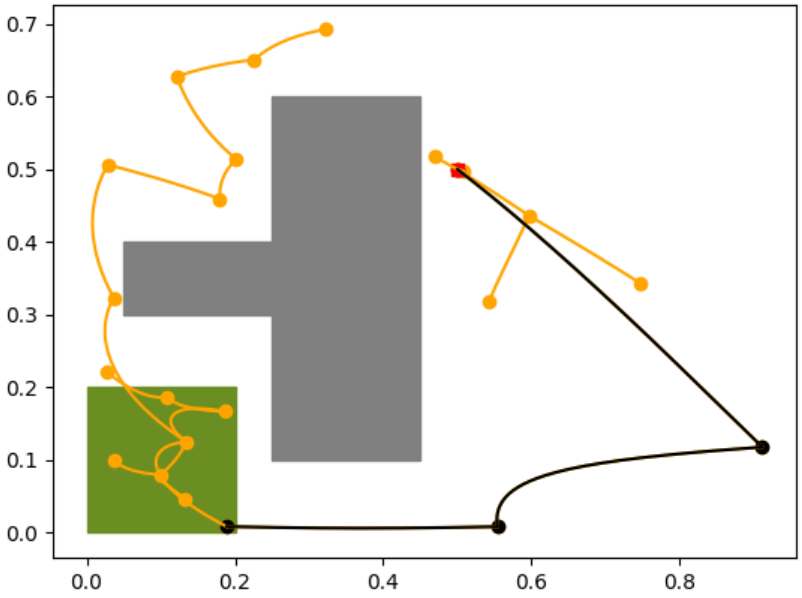

Рис. 5: KRRT*, $N = 250$

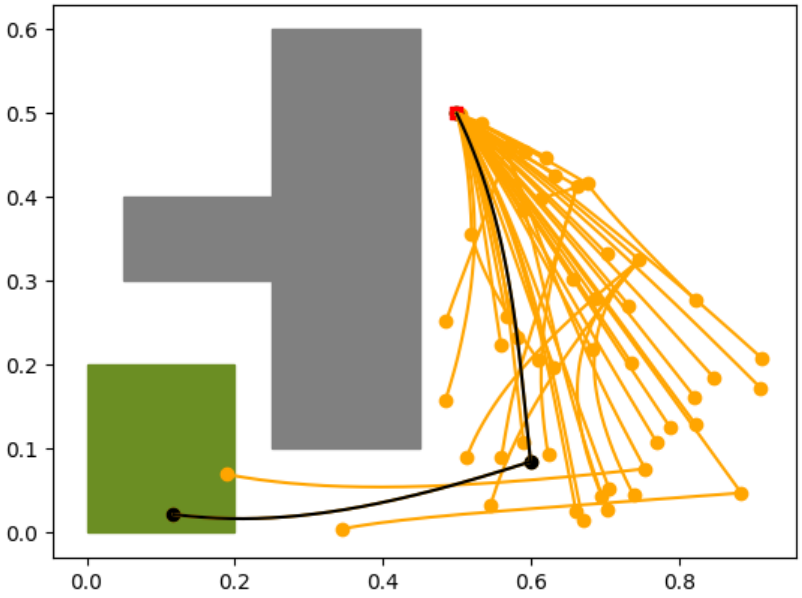

Рис. 6: MyKRRT*, $N = 250$

Поитогу можно сделать вывод о том, что на данном примере KRRT* лучше исследует пространство, однако MyKRRT* добавляет больше вершин к графу. RRT* же ожидаемо превосходит своих соперников в степени дискретизации пространства.

6 Заключение

В результате проделанной работы можно сделать следующие выводы:

1. При правильно подобранных параметрах l, T_{max} предлагаемая модификация алгоритма исследует пространство также широко, как и базовая реализация `KRRT`*, но при этом делает это с большей степенью дискретизации.

2. По времени `MyKRRT`* работает дольше, чем `KRRT`*, однако, если сопоставить увеличение времени работы росту числа вершин в графе, то получим повышение эффективности алгоритма в среднем в 1.1 раз (при инициализации параметров случайным образом), а в отдельных случаях, если удалось правильно подобрать $l(t)$, T_{max} — в два и более раз.

3. Тривиальный алгоритм `RRT`* оказался самым эффективным в плане исследования среды, однако является слабоприменимым, так как неприспособлен к дифференциальным ограничениям на траектории, и за функционал берёт лишь длину пути.

4. Аналитическое разрешение недостатков алгоритма даёт большее ускорение, нежели их программная оптимизация. Моя реализация функции рулевого управления `Steer` слишком долго строила траектории из-за того, что в ней использовался большой и неэффективный перебор по конечному времени. Однако впоследствии был найден способ определять подозрительные на оптимальность конечные моменты через принцип максимума Понтрягина, что позволило уменьшить среднее время работы функции с 1.4s до 20ms.

5. Если говорить о дальнейшем продолжении данной работы, то можно попробовать подобрать эвристику для функции `Nearest`, учитывающую функционал. Кроме того, было бы интересно применить алгоритм к нелинейным системам, линеаризовав их, посмотреть на качество работы, сравнить с аналогами для нелинейных систем.

Список литературы

[1] Paden B., Cap M., Yong S. Z., Yershov D., Frazzoli E., "A survey of motion planning and control techniques for self-driving urban vehicles", IEEE, 2016.

[2] Karaman S., Frazzoli E., "Sampling-based algorithms for optimal motion planning", The International Journal of Robotics Research, vol. 30, pp. 846–894, 2011.

[3] Karaman S., Frazzoli E., "Optimal kinodynamic motion planning using incremental sampling-based methods", IEEE, 2010.

[4] Muller M. E., "A note on a method for generating points uniformly on ndimensional spheres". Comm. Assoc. Comput. Mach. 1959.

[5] Lavretsky E. and Wise. K. "Robust and adaptive control with aerospace applications". London, UK: Springer-verlag, 2013.

[6] LaValle S., "Planning algorithms", Cambridge University Press, 2006.

[7] Lewis F., Syrmos. V., "Optimal control". John Wiley & Sons, 1995.

[8] Perez A., Platt R. Jr., Konidaris G., Kaelbling L. and Lozano-Perez T., "LQR-RRT*: optimal sampling-based motion planning with automatically derived extension heuristics", IEEE ICRA, 2012.

[9] Webb D., Berg J., "Kinodynamic RRT*: optimal motion planning for systems with linear differential constraints", 2012.

[10] LaValle S., Kuffner J., "Randomized kinodynamic planning IEEE, 1999.