Санкт-петербургский государственный университет

Кафедра математического моделирования энергетических систем

Губин Александр Игоревич

Магистерская диссертация

Построение маршрута с учетом динамических ограничений

Направление 01.04.02 "Прикладная математика и информатика"

Магистерская программа "Математическое и информационное обеспечение экономической деятельности"

Научный руководитель,

к. ф.-м. н.,

доцент,

Погожев С.В.

Санкт-Петербург

2020

Содержание

Введение

Впервые о мобильных роботах заговорили после успешной миссии Лунохода-1 [1], который стал первым успешным планетоходом, предназначенным для исследования поверхности луны. Он был доставлен на поверхность Луны 17 ноября 1970 года на борту посадочного модуля Луна-17. Им управляли специально обученные операторы удаленного контроля с Земли. Луноход-1 преодолел расстояние около 10 километров.

В настоящее время, благодаря развитию информационных технологий, мобильные роботы используются не только для космических миссий, а также и в различных отраслях деятельности человека. Например, мобильные роботы используются для погрузки и разгрузки товара на складе. Благодаря тому, что расположение стеллажей на складе не меняется, робот может спокойно маневрировать между ними, не создавая аварийных ситуаций.

Известны примеры использования роботов для обеспечения безопасности человека. В Японии распространены, так называемые, роботы-полицейские, которые движутся по городу и следят за соблюдением порядка на улицах. Они оснащены детекторами дыма, способны издавать сигнал тревоги, а также оснащены средствами видеосъемки.

В 1989 году [2] был изобретен первый автономный робот Helpmate, предназначенный для решения логистических и транспортных задач в больницах. Навигация Helpmate осуществлялась на основе одометрических данных, а с помощью ультразвуковых датчиков измерения расстояния, датчиков инфракрасного излучения и видеокамер робот был способен избегать столкновений с препятствиями. Он использовался для доставки еды и лекарств пациентам в палаты, перевозки документов среди медперсонала больницы.

Компания Meituan Dianping, которая первоначально выступила с инициативой «бесконтактной доставки» по Китаю, уже начала использовать автономные транспортные средства для поставок продуктов в районе Шуньи

в Пекине и планирует запустить аналогичные службы доставки роботов в других районах столицы [3]. Компания начала тестировать роботов и беспилотников для доставки в 2019 году, но это первый случай развертывания автономных средств доставки на дорогах общего пользования. Автономный робот способен перевозить до 100 кг товаров и доставлять от трех до пяти заказов за каждую поездку.

Также широкое распространение имеют роботы, предназначенные для работы в среде, опасной для жизни человека. Например, роботов используют при устранении последствий техногенных катастроф, для работы в местах с повышенным уровнем радиации, для участия в противотеррористических операциях.

Роботы используются для диагностирования, фрезерования и заделки трубопроводов изнутри. Самоходные роботы позволяют быстро находить проблемы, получать достоверную информацию о текущем состоянии трубопроводов, принимать правильное решение о способе ремонта и объеме предстоящих работ.

Построение маршрута движения является одной из важнейших задач в навигации роботов. В основе решения данной задачи лежат три главных условия. Первое – построенный маршрут должен проходить через заданные начальные и конечные точки. Второе – путь робота необходимо строить таким образом, чтобы он не пролегал через ограничения, такие как различные архитектурные сооружения, препятствия природного характера и так далее. И, наконец, третье – построенный маршрут, отвечающий первым двум условиям, должен быть оптимальным.

Подходы к планированию пути можно систематизировать по разным аспектам. В отношении применения информационных технологий, методы возможно разделить на традиционные и эвристические. По имеющейся информации об окружающей среде допускается разделение на подходы с глобальным планированием маршрута (в прямом доступе есть карта местности) и локальным планированием (в наличии сведения об обстановке в

области прямой видимости робота). В зависимости от характера окружающего мира можно классифицировать задачи планирования с учетом динамических и статических ограничений (стоит сказать, что при решении практических задач, только статические ограничения почти не встречаются).

В данной работе ставятся задача построения маршрута с учетом динамических ограничений, а также разрабатывается алгоритм для ее решения. Для иллюстрации работы алгоритма приводятся смоделированные примеры.

Постановка задачи

Дана пара точек на плоскости – начальная A(x_A, y_A) и конечная B(x_B, y_B) точки маршрута. Пусть Ω_0 – область движения объекта. Ω_0 представляет собой прямоугольник, включающий точки A и B.

$$\Omega_0 = \begin{cases} (x,y)|\underline{x} \le x \le \overline{x}, \underline{y} \le y \le \overline{y}, \\ \underline{x} \le x_A \le \overline{x}, \underline{y} \le y_A \le \overline{y}, \underline{x} \le x_B \le \overline{x}, \underline{y} \le y_B \le \overline{y} \end{cases}$$

Область ограничений Ω_1 представляет собой однопараметрическое (параметром является время) семейство:

$$\Omega_{1i}(t), \qquad i = \overline{1,N}, \qquad t = [T_0, T_1],$$

где T_0 - время начала пути, а T_1 – время прибытия в конечную точку.

Тогда $\Omega_{2j}(t)$ – допустимая область движения выглядит следующим образом:

$$\Omega_{2j}(t) = \Omega_0 \setminus \bigcup_{i=1}^{N} \Omega_{1i}$$

Необходимо построить маршрут наименьшей длины, полностью лежащий в допустимой области движения $\Omega_{2j}(t)$.

Глава 1. Обзор существующих методов построения маршрута

В этой главе рассмотрены три подхода к построению маршрута, а именно методы на основе графов, методы на основе клеточной декомпозиции и методы на основе потенциальных полей. Все эти алгоритмы в изначальном своем виде предназначены только для работы со статическим окружением, то есть со средой, где присутствуют только статические ограничения.

1.1. Алгоритмы на основе графов

Для начала следует вспомнить, что графом G называется множество вершин V и набор E пар вершин, соединённых ребром. Обозначается G=(V,E). Ребра могут иметь или не иметь направление (направленное ребро называется дугой), а граф называется ориентированным (если все ребра в графе имеют направление) и не ориентированным, в противном случае. Также каждому ребру может быть поставлено в соответствие некоторое значение (вес). В таком случае граф называется взвешенным.

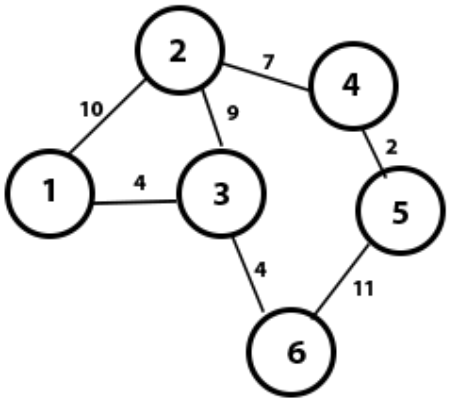

Рис. 1. Взвешенный граф

Для решения задачи построения пути в среде с ограничениями существует несколько подходов. Одним из них является метод построения графа видимости (Рис. 2) [4].

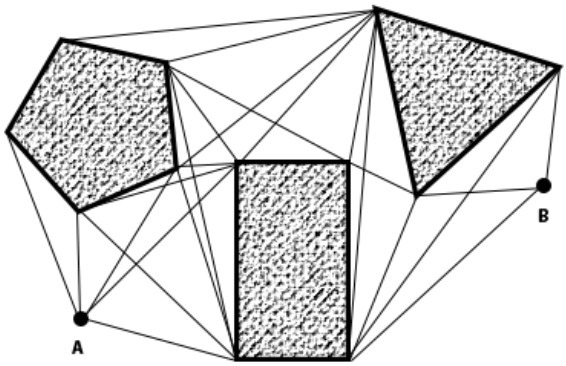

Рис. 2. Граф видимости

Пусть заданы начальная точка (A) и конечная точка (B), также известны ограничения, являющиеся выпуклыми многоугольниками. Тогда граф видимости $Vg(N, Q)$ – неориентированный граф, с множеством вершин $N = W \cup \{A, B\}$, где W – набор всех вершин ограничений, и множеством всевозможных ребер $Q = \{q \mid (n_i n_j), n_i n_j \in N, n_i \neq n_j\}$ таких, что никакое ребро q не пересекает ни одно из ограничений. Собственно, граф видимости называется так, потому что каждые две вершины, соединённые ребром могут друг друга "видеть". Пример кратчайшего маршрута в таком графе представлен стрелками на Рис. 3.

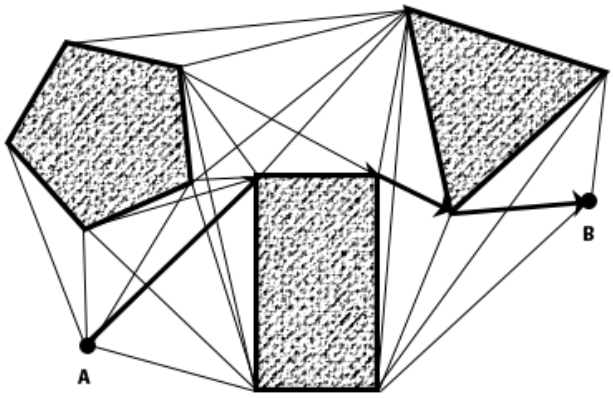

Рис. 3. Кратчайший маршрут в графе $Vg = (N, Q)$, обозначенный стрелками.

Граф видимости достаточно прост в понимании и программной реализации, однако имеет один очень существенный недостаток – при увеличении количества ограничений сильно возрастает трудоемкость вычислений. Поэтому были предложены алгоритмы сокращения графа видимости, такие как динамический граф видимости [5] и необходимый граф

видимости [6].

Общей идеей [5] и [6] является построение только потенциально нужных для маршрута ребер. То есть при анализе существующих ограничений отбрасываются те, которые мало влияют на движение робота.

Другим подходом для решения задачи построения пути является метод, основанный на подходе структурирующего свободного пространства [7], включающий в себя три основных этапа:

1) Формирование структурирующего свободного пространства, на основе начальных данных об ограничениях (Рис. 4а) и свободных связей (free links), то есть отрезков, соединяющих вершины ограничений, при этом, не пересекая никаких других ограничений (Рис. 4б).
2) На втором шаге формируется так называемый граф MAKLINK, вершинами которого являются середины свободных связей (Рис. 4в).
3) На третьем шаге с помощью алгоритмов поиска кратчайшего пути в графе определяется маршрут движения робота (Рис. 4г).

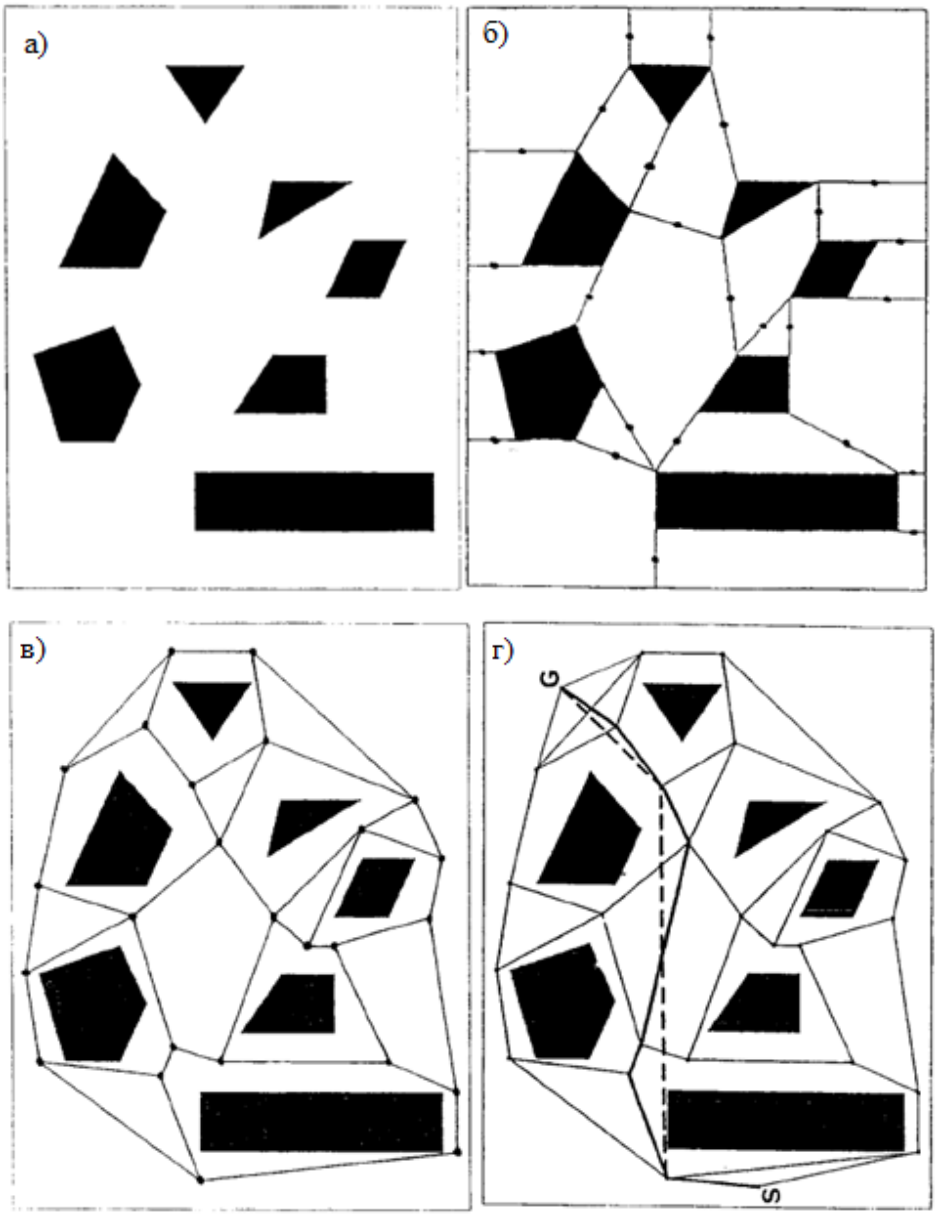

Рис. 4. а) начальная обстановка вокруг робота, содержащая 7 ограничений; б) сформированное структурирующее свободное пространство, на основе свободных связей, с вычисленными серединами каждого отрезка; в) граф MAKLINK; г) пример маршрута, построенного на основе графа MAKLINK. Изображения взяты с [7].

Так же стоит упомянуть подход случайного планировщика пути (randomized path planner, RPP) [8]. Этот метод хорош тем, что ограничения могут не быть выпуклыми многоугольниками и иметь практически любой вид.

На первом этапе этого алгоритма генерируются вершины-кандидаты для будущего графа. Последовательно рассматривается каждое ограничение и на его границе с учетом некой метрики случайным образом выбираются

точки, которые возможно и станут узлами будущего графа. Далее на каждом шаге итерации алгоритма пытаются соединить начальную и конечную точки, и если это не выходит, случайным поиском среди k ближайших соседей конечной точки находят наиболее подходящую. Итерации повторяют до тех пор, пока не будет получен путь от начальной точки до конечной.

1.2. Методы на основе клеточной декомпозиции

При построении маршрута в среде с ограничениями можно пытаться дискретизировать окружающую среду. Если говорить в общем, планирование пути с помощью алгоритмов клеточной декомпозиции представляет собой нахождение последовательности клеток (все клетки образуют связный граф), которые должен пройти робот, чтобы попасть из начальной точки в конечную точку.

Существуют методы приближенной [9] и точной клеточной декомпозиции [10]. Приближенная клеточная декомпозиция реализуется с помощью сетки (grid map), покрывающей пространство (Рис. 5). Каждая клетка сетки имеет одинаковые размеры и содержит в себе значение, которое характеризует ее как свободную или занятую препятствием.

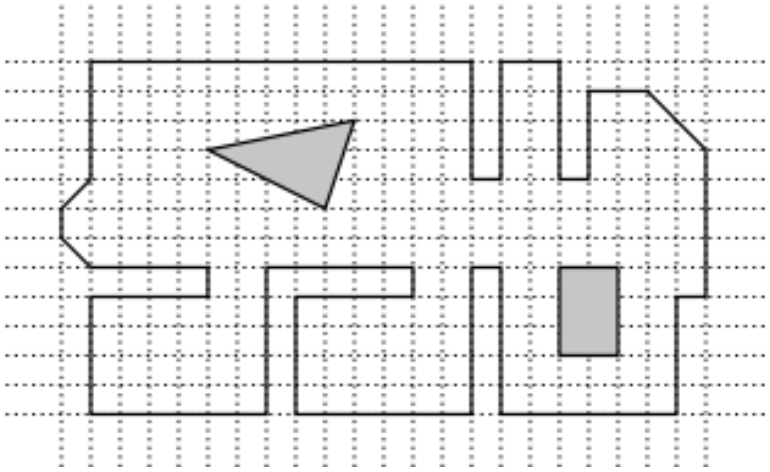

Рис. 5. Приближенная клеточная декомпозиция. Закрашенные многоугольники являются препятствиями. Изображение взято с [9].

Сетка достаточно легка в использовании и проста в программной реализации. Однако имеется один большой минус – при увеличении разрешения сетки (уменьшении размера клетки), возрастает трудоемкость вычислений. Предложены разные способы снизить объем вычислений с использованием нерегулярных сеток, где самым известным является дерево квадрантов (quadtree) [11].

Точная клеточная декомпозиция дискретизирует пространство в клетки, полностью свободные от ограничений (Рис. 6). Согласно [12], клетки,

полученные после точной клеточной декомпозиции, должны удовлетворять двум условиям:

1) Форма каждой ячейки должна быть достаточно простой, чтобы вычисление пути между любыми двумя клетками не было трудоемким
2) Проверка соседства двух клеток не должна быть сложной.

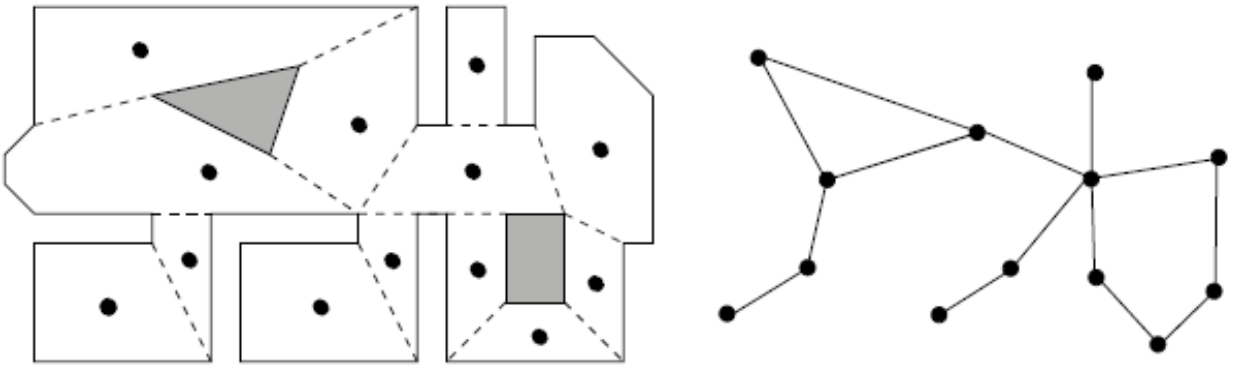

Рис. 6. Точная клеточная декомпозиция (слева) и соответствующий ей граф (справа). Изображение взято с [9].

Самым хорошо изученным методом точной клеточной декомпозиции является вертикальная или трапециевидная декомпозиция [13] (Рис. 7).

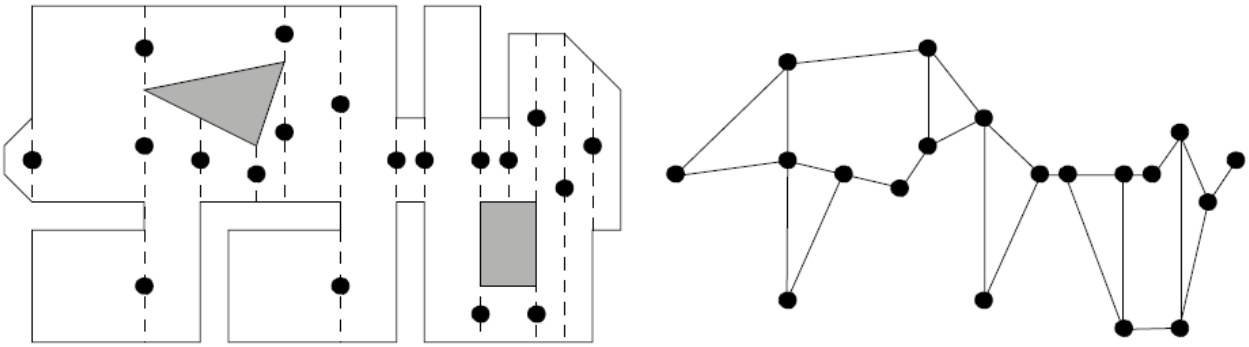

Рис. 7. Вертикальная клеточная декомпозиция (слева) и соответствующий ей граф (справа). Изображение взято с [9].

При решении задачи нахождении маршрута с помощью клеточной декомпозиции, можно использовать разные методы нахождения пути в графе. Их можно поделить на классические (поиск в ширину, поиск в глубину, поиск с ограничением глубины, алгоритм Дейкстры и так далее) и эвристические.

Эвристические стратегии поиска не имеют под собой строго

математического обоснования, однако при решении практических задач дают приемлемые результаты. Особенностью эвристических алгоритмов являются эвристические правила, которые и позволяют повысить эффективность алгоритма в сравнении с соответствующей классической стратегией.

Классическим примером эвристического алгоритма поиска пути является алгоритм A*, предложенный Н. Нильсоном, П. Хартом и Б. Рафаэлем в 1968 году [14]. A* представляет собой расширение алгоритма Дейкстры. Целевая функция алгоритма A* $f(n)$ выглядит следующим образом:

$$f(n) = g(n) + h(n),$$

где $g(n)$ – функция стоимости достижения конечной точки из начальной точки, а $h(n)$ – эвристическая функция оценки расстояния из начальной точки в конечную точку, которая не должна переоценивать расстояние до конечной точки.

Так A* получило продолжение в виде алгоритмов D*[15] и Theta*[16]. Алгоритм D* (динамический A*) представляет собой продолжение алгоритма A*. Метод D* является более эффективным, поскольку в нем не планируется весь путь до конечной точки. Алгоритм D* в дальнейшем получил продолжение в виде алгоритмов Lifelong Planning A* [17] и D* Lite [18].

Также к способам нахождения пути на основе клеточной декомпозиции можно отнести метод распространения волнового фронта, в частности алгоритм the Fast Marching Method (FMM) [19]. Он основывается на том, что, согласно принципу Ферма, луч света всегда проходит из начальной точки в конечную по пути наименьшей длины. Математически, распространение света выражается уравнением эйконала (Eikonal equation).

Предположим, что плоскость покрыта сеткой, задана точка $A = (x, y)$ с прямоугольными координатами, $D(A)$ – функция времени прибытия волны в конечную точку, $W(A)$ – скорость распространения света. Также мы

предположим, что свет начинает распространяться во время $D = 0$ со скоростью W всегда неотрицательной. Уравнения эйконала определяет время прибытия распространяющейся волны $D(A)$ в каждой точке A, в которой скорость распространения определена, $W(A)$. То есть:

$$|\nabla D(A)|W(A) = 1.$$

Согласно [20], дискретизируя градиент ∇D, возможно решить уравнение эйконала в каждой точке $B(x_i, y_j)$, где i и j – строка и колонка сетки:

$$D_1 = min(D_{i-1,j}, D_{i+1,j})$$

$$D_2 = min(D_{i,j-1}, D_{i,j+1})$$

$$\left(\frac{D_{i,j} - D_1}{\Delta x}\right)^2 + \left(\frac{D_{i,j} - D_2}{\Delta y}\right)^2 = \frac{1}{W_{i,j}^2}$$

FMM состоит в том, что $D_{i,j}$ находится для каждой ячейки сетки, начиная с точки распространения света $D_{i_0,j_0} = 0$. В следующей итерации считаются $D(i,j)$ для соседей точек из предыдущей итерации. В конечном итоге, с помощью градиентного спуска построить путь из любой точки к источнику распространения волны, которая в рамках задачи является конечной точной маршрута. Главное преимущество данного подхода состоит в том, что полученный путь является оптимальным (Рис. 8).

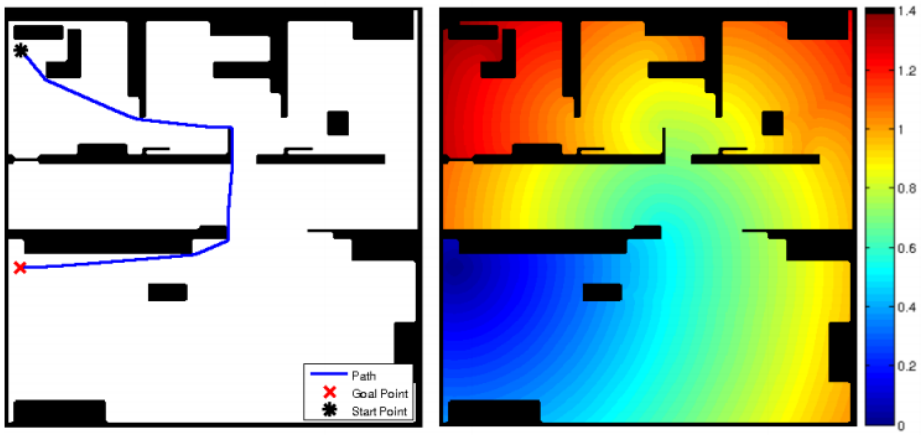

Рис. 8. Пример пути, полученный с помощью метода the Fast Marching Method. Слева показана оригинальная плоскость и построенный путь. Справа карта расстояний, рассчитанная the Fast Marching Method. Изображение взято с [12].

1.3. Методы на основе потенциальных полей

Самым известным представителем данного семейства алгоритмов является метод искусственных потенциальных полей (Artificial potential field, APF) [21]. Его главная идея заключается в том, что робота помещают в искусственное потенциальное поле, в котором робот притягивается к конечной точке и отталкивается от встречающихся ограничений.

Обозначим позицию робота за $q = (x, y)$. Тогда притягивающий потенциал примет вид:

$$U_{att}(q) = \frac{1}{2}\,\delta s^m\big(q, q_{finish}\big),$$

где δ – некоторый позитивно возрастающий фактор, $s\big(q, q_{finish}\big) = \big\|q_{finish} - q\big\|$ – расстояние между роботом (q) и конечной точкой и $m = 1$ или 2. Если $m = 1$, то притягивающий потенциал имеет коническую форму и тогда притягивающая сила имеет постоянную амплитуду, за исключением конечной точки, где U_{att} сингулярный. Если $m = 2$, то притягивающий потенциал имеет параболическую форму и соответствующая притягивающая сила представляет собой отрицательный градиент притягивающего потенциала:

$$F_{att}(q) = -\nabla U_{att}(q) = \delta\big(q_{finish} - q\big),$$

которая линейно сходится к 0 по мере приближения робота к конечной точке.

Отталкивающий потенциал имеет следующий вид:

$$U_{rep}(q) = \begin{cases} \frac{1}{2}\gamma\left(\frac{1}{s(q, q_{obs})} - \frac{1}{s_0}\right)^2, & \text{если } s(q, q_{obs}) \le s_0 \\ 0, & \text{если } s(q, q_{obs}) > s_0 \end{cases},$$

где γ – некоторый позитивно возрастающий фактор, $s(q, q_{obs})$ определяет минимальное расстояние от робота (q) до ограничения, q_{obs} – такая точка на ограничении, что расстояние от нее до робота и есть минимальное расстояние роботом и ограничением. Соответствующая отталкивающая сила имеет следующую форму:

$$F_{rep}(q) = -\nabla U_{rep}(q)$$

$$= \begin{cases} \gamma\left(\frac{1}{s(q,q_{obs})} - \frac{1}{s_0}\right)\frac{1}{s^2(q,q_{obs})}\nabla s(q,q_{obs}), \text{если } s(q,q_{obs}) \le s_0 \\ 0, \qquad\qquad \text{если } s(q,q_{obs}) > s_0 \end{cases}.$$

Общая сила, воздействующая на робота и определяющая его движение, является суммой притягивающей и отталкивающей сил:

$$F_{total} = F_{att} + F_{rep}$$

Данный метод привлекателен благодаря своей математической элегантности и простоте, однако имеет ряд серьезный недостатков [22]:

1) Возникают ситуации локального минимума
2) Колебания вблизи ограничений
3) Робот не может пройти между двумя близкорасположенными ограничениями
4) В ситуациях, когда конечная точка находится на границе ограничения, для робота по мере приближения притягивающая сила слабеет, а отталкивающая, наоборот, возрастает. В результате чего робот не может в конечную точку.

Были предложены решения проблемы №1 [23-28], которые можно классифицировать на две категории: избегание локального минимума (local minima avoidance, LMA) и побег из локального минимума (local minima escape, LME). Однако все предложенные подходы сильно увеличивают трудоемкость вычислений, что их использование в задачах построения маршрута в реальном времени не представляется возможным.

Решение проблемы №2 предложено в [29]. Метод виртуального силового поля (virtual force field, VFF). Но и у данного подхода имеется серьезный недостаток: возникает проблема №3. Поэтому в 1991 году был представлен метод гистограммы векторного поля (the vector field histogram, VFH) [30]. Алгоритм VFH является локальным планировщиком пути. То есть он не пытается найти оптимальный маршрут (найти оптимальный маршрут

возможно, только если дана полная информация об окружающей среде). Метод VFH работает в три этапа:

1) Строится двумерная гистограмма C в прямоугольной системе координат, содержащая полную информацию об окружении робота. Обновляется в реальном времени по мере поступления новой информации об ограничениях.
2) Строится одномерная гистограмма H в полярной системе координат, которая содержит кратковременную информацию вокруг робота. Гистограмма содержит n угловых секторов с градусной мерой φ. В результате трансформации каждый сектор k из H содержит значение h_k, показывающее полярную плотность ограничений в направлении, за которое отвечает сектор k.
3) Роботу подаются выходные значения для движения.

Стоит отметить, что алгоритм VFH не решает проблему возникающих локальных минимумов, поэтому в конечном итоге робот может начать топтаться на месте и не достичь конечной точки.

Глава 2. Алгоритм построения маршрута

2.1. Описание алгоритма

В рассмотренной ранее главе были описаны три основных подхода к построению маршрута с динамически меняющимися ограничениями. Для выбора наиболее подходящего метода был проведен анализ основных преимуществ и недостатков каждого из них.

Так, методы на основе графов достаточно эффективно работают со статичным окружением и с динамически меняющимся окружением, о котором известно все в каждый момент процесса построения маршрута. Однако, в реальности невозможно иметь четкий прогноз о возникающих в процессе ограничениях. Вдобавок к этому, при увеличении количества ограничений, трудозатраты на построение пути существенно увеличиваются.

Методы на основе потенциальных полей хорошо работают со статическим окружением, к тому же количество ограничений влияет на трудоемкость вычислений незначительно. К тому же, длина полученного маршрута движения всегда будет оптимальной. Однако, возникающие ситуации локального минимума, проблема колебаний вблизи ограничений, а также возможные ситуации, когда конечная точка будет вблизи ограничений, заставляют задуматься о выборе данной группы методов.

И наконец, методы на основе клеточной декомпозиции. Эти алгоритмы прекрасно работают с окружением любого характера, они просты в реализации. Время работы таких алгоритмов сильно зависит масштаба сетки, однако предложены эффективные методы, позволяющие сократить трудозатраты при клеточной декомпозиции. К недостаткам можно отнести то, что маршрут не всегда будет являться оптимальным.

Исходя из анализа рассмотренных алгоритмов, было принято решение разработать алгоритм на основе клеточной декомпозиции.

Рассмотрим разработанный алгоритм решения задачи построения

маршрута с учетом динамических ограничений:

1. Отметить на плоскости начальную $A(x_A, y_A)$ и конечную $B(x_B, y_B)$ точки маршрута.
2. Разбить плоскость на клетки ($v \in V$), по которым и будет осуществляться движение робота. Отметить клетки, которые содержат точки A и B (v_{start} и v_{finish} соответственно). Функцию эвристической оценки расстояния между вершинами v_1 и v_2 обозначим $p(v_1, v_2)$.
   2.1 Каждая клетка сетки содержит значение $g(v)$, которое обозначает стоимость достижения рассматриваемой клетки v из v_{finish}.

$$g(v) = \begin{cases} 0, & \text{если } v = v_{finish} \\ min_{\hat{v} \in parents(v)}(g(\hat{v}) + p(v, \hat{v})) \end{cases},$$

   где $parents(v)$ – соседние клетки, из которых мы можем попасть в клетку v. У каждой клетки имеется 8 соседей. Значение $g(v)$ каждой клетки сетки при инициализации равняется $+\infty$.
3. Вычисляем $g(v)$ для элементов сетки, последовательно двигаясь от v_{finish} к v_{start} таким образом, что:
   3.1. Вычисляем значение $g(v_{finish})$.
   3.2. Вычисляем значения $g(\hat{v})$, где $\hat{v} \in children(v_{finish})$, $children(v)$ – соседние клетки, в которые мы можем попасть из клетки v.
   3.3. Выбираем среди $children(v_{finish})$ такую вершину $\tilde{v}$, для которой значение эвристической функции оценки расстояния будет минимальным.
   3.4. Повторяем шаг 3.2, только вместо v_{finish} теперь будет $\tilde{v}$.
   3.5. Пусть $\check{v}$ – вершина, из которой мы попали в $\tilde{v}$. Если окажется, в шаге 3.3 не найдены подходящие вершины, то возвращаемся к вершине $\check{v}$ и повторяем шаг 3.3, только

выбираем мы теперь среди вершин $children(\check{v})\backslash\tilde{v}$.

3.6. В момент попадания в вершину v_{start} прекращаем вычисления – все нужные значения $g(v)$ мы уже нашли.

4. Начинаем строить маршрут, двигаясь от клетки v_{start}, выбирая среди $children(v)$ клетку с наименьшим значением g:

$$g(v_{next}) = min_{\hat{v} \in children(v)}(g(\hat{v})).$$

Таких клеток может быть несколько. В таком случае мы выбираем нужную исходя из эвристической функции оценки расстояния $p(v_{next}, v_{finish})$.

5. При встрече с клетками, содержащими препятствие, меняем значение g этих клеток на $+\infty$.
6. В ситуации, когда $g(v_{current}) \leq g(\hat{v})$, где $\hat{v} \in children(v_{current})$, $v_{current}$ – текущая точка пути, возвращаемся к шагу 3, только в роли v_{start} будет выступать $v_{current}$.
7. В момент, когда $v_{current} = v_{finish}$, заканчиваем движение – маршрут построен.

Стоит отметить, что разработанный алгоритм никак заранее не учитывает местонахождение ограничений при построении маршрута. Корректирование пути происходит только при непосредственном контакте робота с ограничением, что позволяет сказать о том, что предложенный метод способен давать приемлемый результат в полностью неизвестной среде движения.

Разработанный метод отличается от стандартного метода клеточной декомпозиции (например A* [14]) тем, что расчет значений $g(v)$ происходит только для нужных клеток (См. пункт 3 алгоритма), что позволяет существенно сократить объем производимых вычислений, а также тем, что в алгоритме есть шаг перепланировки (См. пункт 6 алгоритма), что позволяет построить маршрут в среде с динамическими ограничениями.

Сравним скорость работы и процент использованных клеток трех

алгоритмов на основе клеточной декомпозиции, а именно A*[14], D*[15] и предложенный в данной работе алгоритм (Рис. 9).

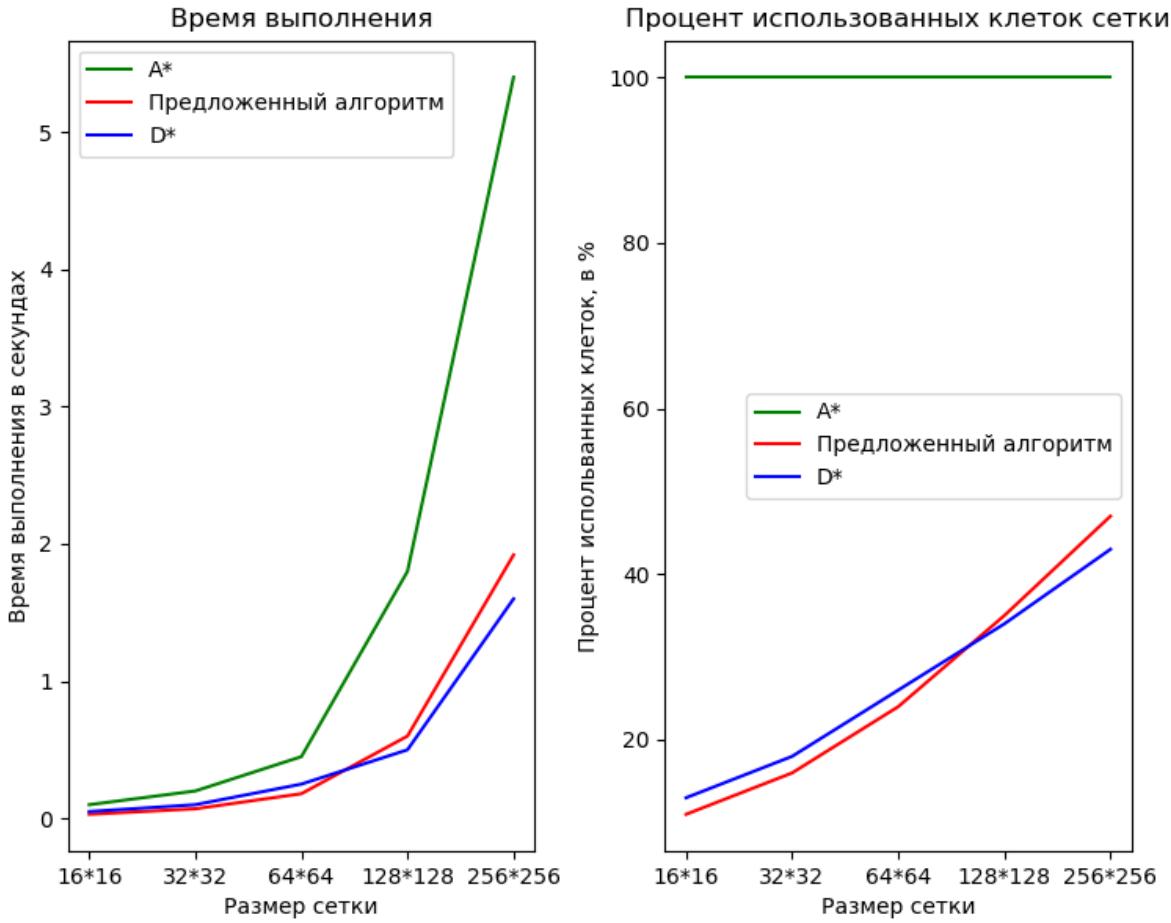

Рис. 9. Сравнение времени выполнения и процента использованных клеток трех алгоритмов: A*, D* и предложенный в работе алгоритм.

Как видно из иллюстрации выше, предложенный метод работает быстрее и использует меньше клеток при расчетах, чем метод A*. Однако, алгоритм D* работает немного эффективнее на сетках размерами $128 * 128$ и $256 * 256$ клеток, поскольку на шаге вычисления значений сетки в нем используется очередь с приоритетом [15]. Но не стоит забывать, что алгоритмы A* и D* не работают со средой, содержащей динамические ограничения.

2.2. Анализ работы алгоритма

Для программной реализации алгоритма был выбран язык программирования Python 3. Для моделирования была использована библиотека pygame [31], которая, в свою очередь, реализована на языках программирования C/C++, что обеспечивает высокую производительность. Полный исходный код программы можно посмотреть на github [32].

Для иллюстрации работы алгоритма рассмотрим несколько возможных ситуаций.

Пример №1 (Рис. 10), когда окружение имеет только статические ограничения. Размер сетки составляет $16 * 16$ клеток. Начальная точка A движения имеет координаты $(43, 64)$, конечная точка B имеет координаты $(381{,}401)$. Клетка v_{start}, в которой содержится точка A, отмечена оранжевым цветом, а клетка v_{finish}, в которой содержится точка B, отмечена зеленым цветом. Статические ограничения, которые не были учтены роботом, обозначены серым цветом, а ограничения, повлиявшие на маршрут, обозначены черным цветом. Положение робота обозначено красным кружком.

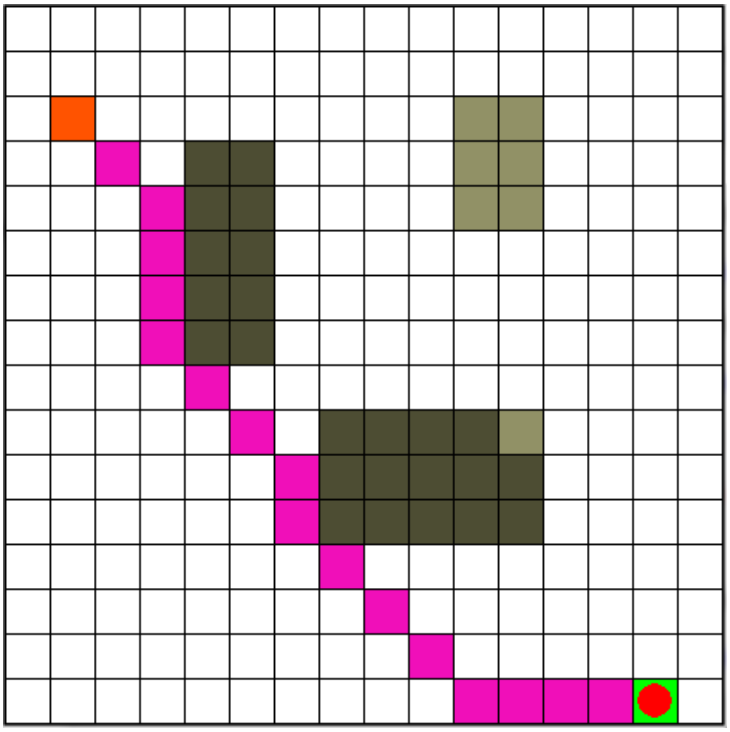

Рис. 10. Пример работы алгоритма со статической средой.

Как видно из иллюстрации выше, робот достиг конечной точки, успешно избегая встреченные на пути препятствия. Построенный маршрут обозначен фиолетовым цветом.

Пример №2 (Рис. 11), когда окружение имеет не только статические, но и динамические ограничения. Размер сетки составляет $32 * 32$ клетки. Начальная точка движения A имеет координаты $(19,398)$, конечная точка B имеет координаты $(368,84)$. В момент нахождения робота в точке C (обозначена крестиком) с координатами $(265,177)$ появилось динамическое ограничение. Динамические ограничения, которые не были учтены роботом, обозначены голубым цветом, а ограничения, повлиявшие на построенный путь, обозначены синим цветом.

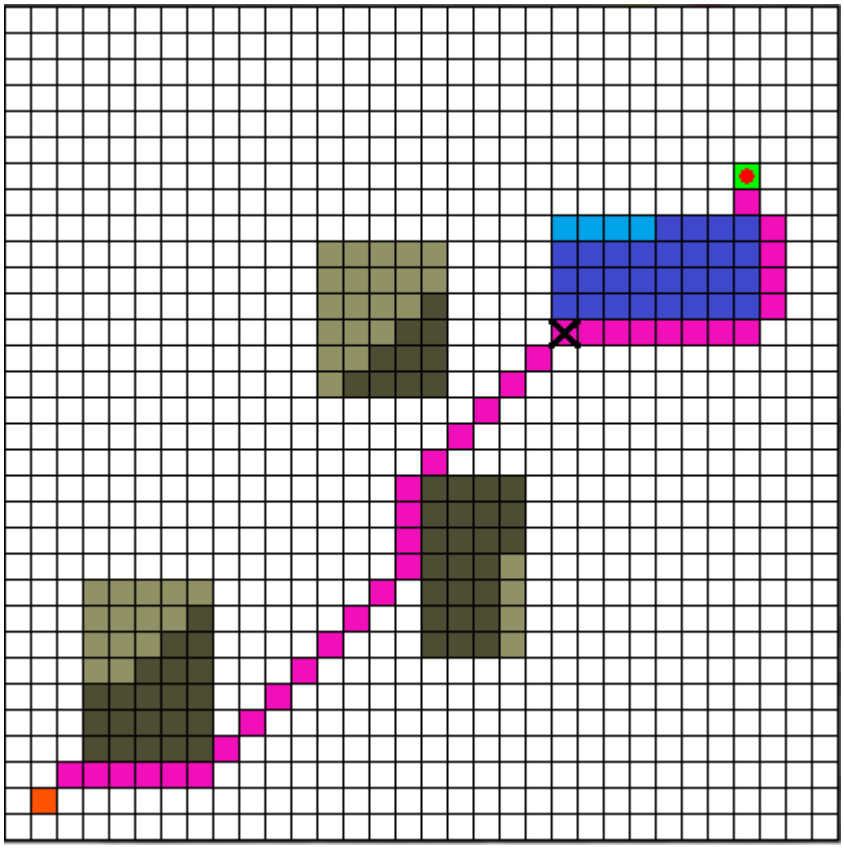

Рис. 11. Построенный маршрут с учетом статических и динамических ограничений

Как видно из Рис. 11, робот эффективно справился с возникшими динамическими ограничениями и попал в конечную точку.

Пример №3 (Рис. 12), когда возникает ситуация, описанная в пункте 6

алгоритма. Размер сетки составляет $16 * 16$ клеток. Начальная точка A имеет координаты $(120{,}401)$, конечная точка B имеет координаты $(377{,}40)$. В момент нахождения робота в точке C с координатами $(299, 221)$ (на рисунке робот как раз находится в клетке $v_{current}$, которая содержит точку C), появилось динамическое ограничение. Синими числами обозначены значения $g(v)$, $g(v\_current) = 7$. Стоит напомнить, что при инициализации все клетки сетки имеют значение $g(v)$ равное $+\infty$, в пустых клетках это значение осталось без изменений. Как видно из рисунка $g(v_{current}) \leq g(\hat{v})$, где $\hat{v} \in children(v_{current})$, поэтому требуется пересчет значений $g(v)$ сетки. Результат пересчета значений показан на Рис. 13.

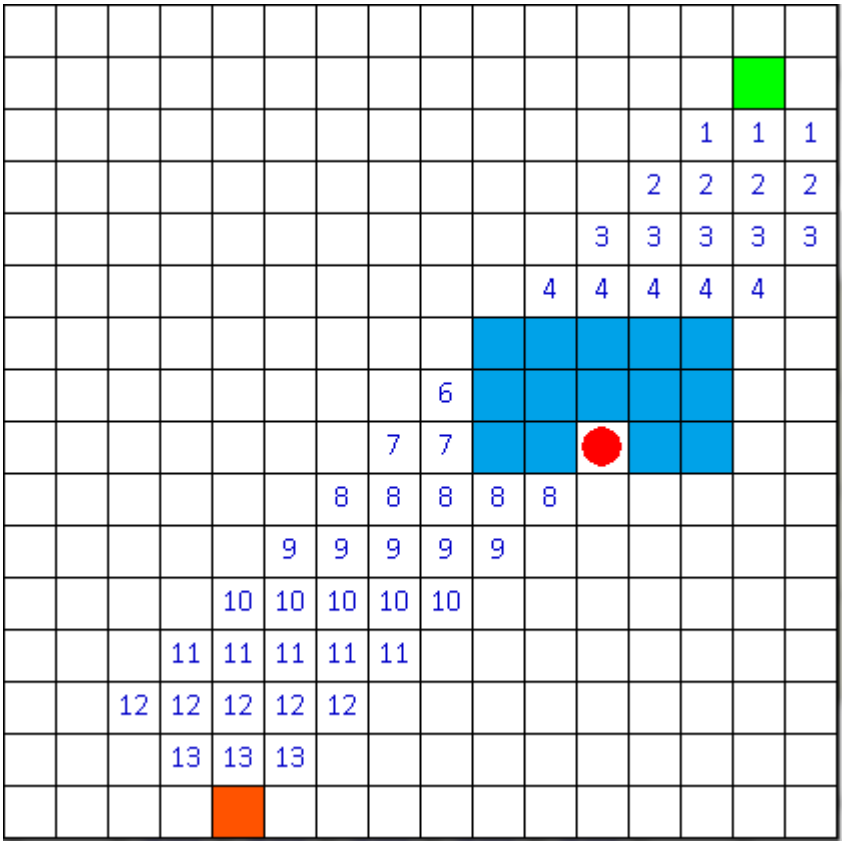

Рис. 12. Ситуация, когда требуется пересчет значений $g(v)$ сетки.

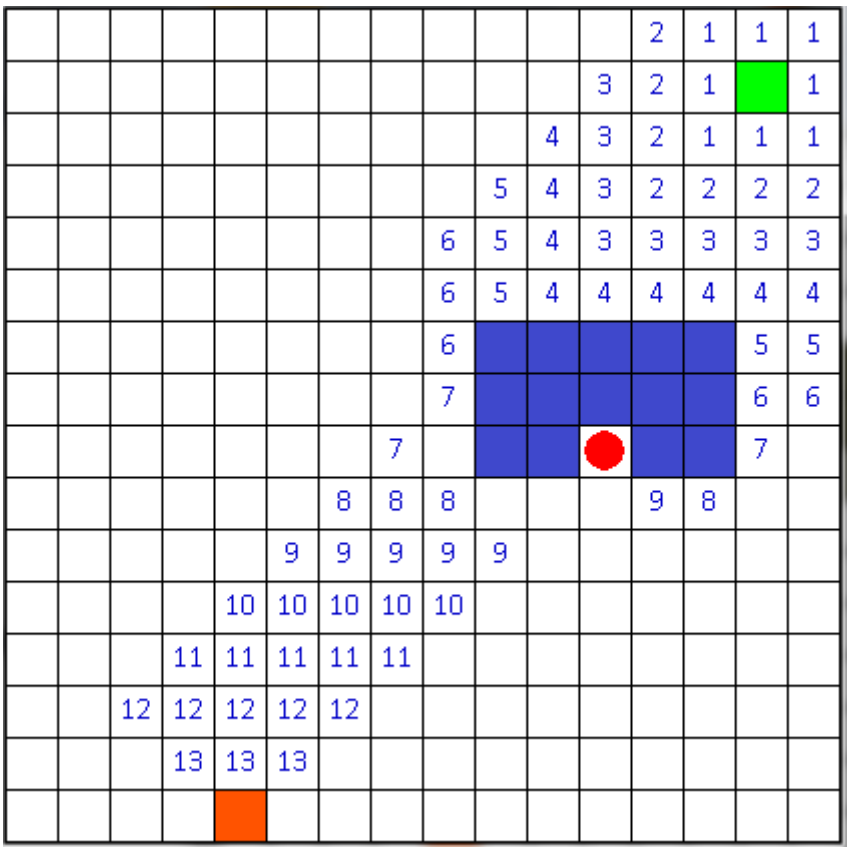

Рис. 13. Полученные после пересчета значения сетки.

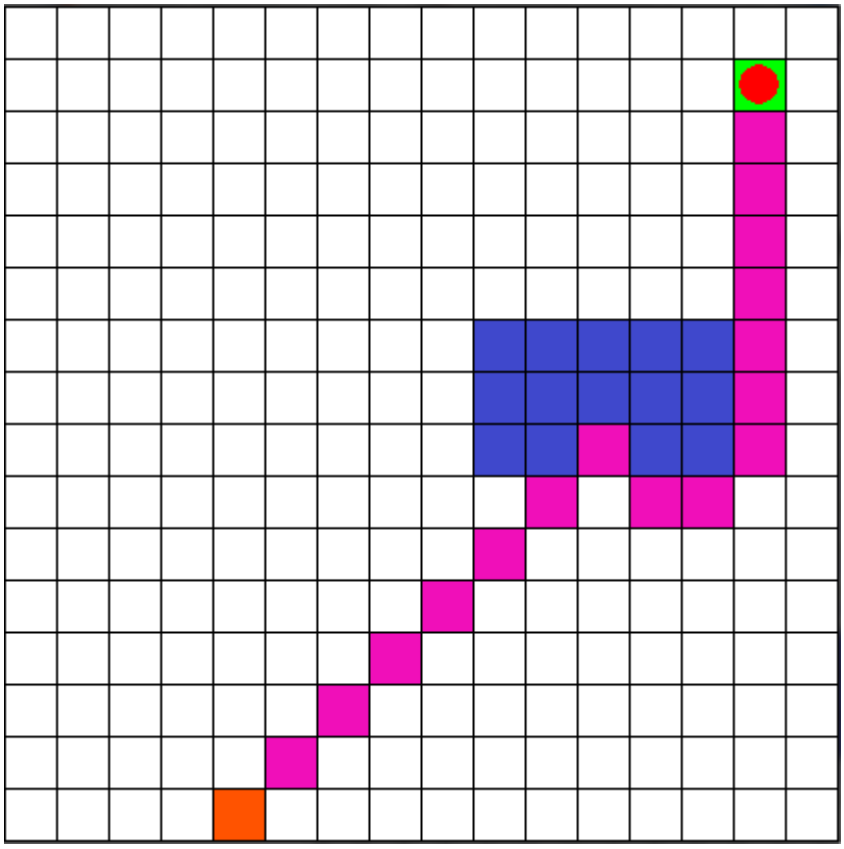
Рис. 14. Полученный в результате маршрут.

На Рис. 14 видно, что алгоритм справился с возникшими

ограничениями и необходимостью пересчета значений сетки, а также построил оптимальный маршрут движения.

Пример №4 (Рис. 15), когда размер сетки большой, к тому же имеется большое количество ограничений, как статических, так и динамических. Размер сетки составляет $150 * 150$ клеток. Начальная точка A имеет координаты $(14{,}544)$, конечная точка B имеет координаты $(594, 15)$. С помощью библиотеки random [33] были сгенерированы 3000 статических ограничений и 2000 динамических ограничений, возникающих в определенные моменты движения робота.

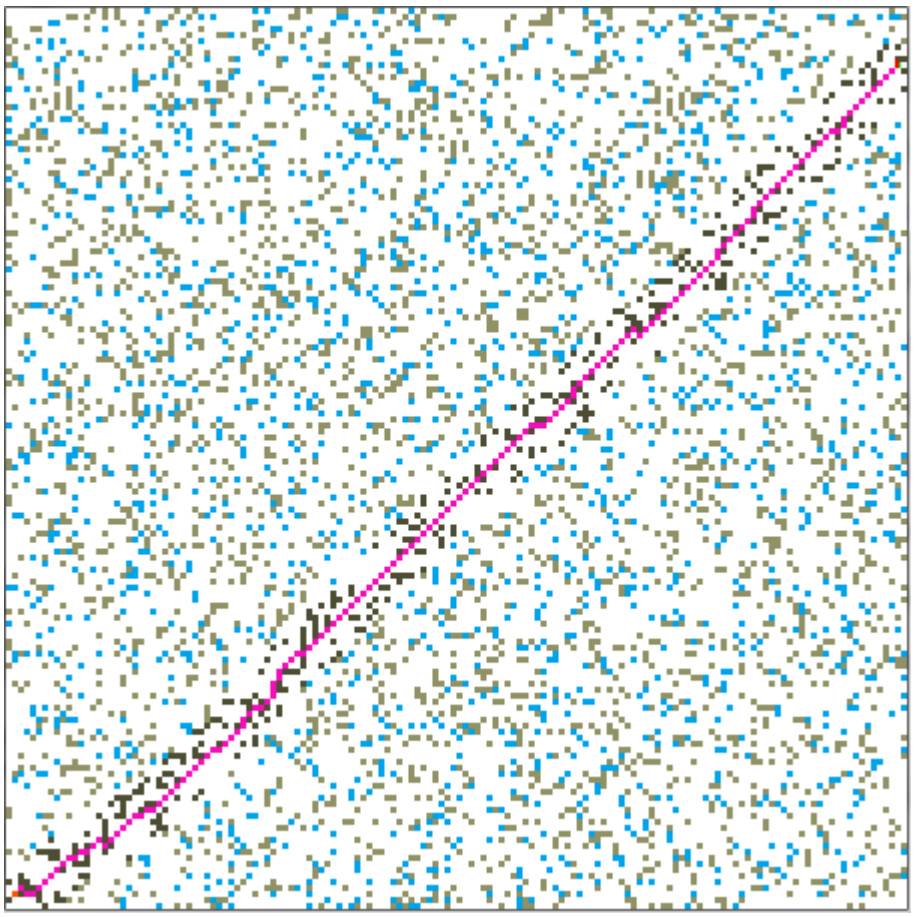

Рис. 15. Пример работы алгоритма на сетке размером 150*150 клеток, содержащей 3000 статических ограничений и 2000 динамических ограничений.

Как видно из Рис. 15, алгоритм справляется и с такой задачей, позволяя построить сложный маршрут.

Заключение

В работе рассмотрены основные подходы к решению задачи построения маршрута с динамически меняющимися ограничениями, а именно методы на основе графов, алгоритмы на основе клеточной декомпозиции и подходы на основе потенциальных полей. Проведен анализ рассмотренных методов, а также выбран наиболее оптимальный подход для решения поставленной задачи.

Предложен алгоритм построения маршрута с учетом динамических ограничений, основанный на клеточной декомпозиции.

Предложенный алгоритм реализован на языке программирования Python, с использованием библиотеки pygame для моделирования. Проведено тестирование работоспособности алгоритма в различных ситуациях.

Список литературы

1. История «Лунохода-1» и работа над ошибками. Роскосмос. URL: https://www.roscosmos.ru/26284/
2. John Evans, Bala Krishnamurthy, Will Pong, Robert Croston, Carl Weiman, Gay Engelberger. HelpMate™: A robotic materials transport system. Robotics and Autonomous Systems, 1989.
3. Meituan Dianping offers ‘contactless shields’ that allow people to eat their delivered food in private | South China Morning Post. URL: https://www.scmp.com/tech/apps-social/article/3074644/meituan-dianping-offers-contactless-shields-allow-people-eat-their
4. T. Lozano- Perez and M. A. Wesley, An algorithm for planning collision-free paths among polyhedral obstacles, Contmum. ACM, vol. 22, pp. 560-570, 1979.
5. Han-Pang Huang, Shu-Yun Chung. Dynamic visibility graph for path planning // IEEE-RSJ intern. conf. on intelligent robots and systems: IROS 2004 (Sendai, Japan, Sept. 28 - Oct. 2, 2004): Proc. N.Y.: IEEE, 2004. Vol. 3. Pp. 2813–2818.
6. Janet J.A., Luo R.C., Kay M.G. The essential visibility graph: An approach to global motion planning for autonomous mobile robots // IEEE intern. conf. on robotics and automation (Nagoya, Japan, May 21-27, 1995): Proc. Vol. 2. N.Y.: IEEE, 1995. Pp. 1958–1963.
7. Habib M.K., Asama H. Efficient method to generate collision free paths for an autonomous mobile robot based on new free space structuring approach // IEEE/RSJ intern. workshop on intelligent robots and systems: IROS'91 (Osaka, Japan, November 3-5, 1991): Proc. Vol. 2. N.Y.: IEEE, 1991. Pp. 563–567.
8. Amato N.M., Wu Y. A randomized roadmap method for path and manipulation planning // IEEE intern. conf. on robotics and automation (Minneapolis, USA, April 22-28, 1996): Proc. Vol. 1. N.Y.: IEEE, 1996. Pp. 113–120.
9. Elfes A. Using occupancy grids for mobile robot perception and navigation.

Computer, 1989, vol. 22, no. 6, pp. 46–57.

10. Sleumer N.H., Tschichold-Gurman N. Exact cell decomposition of arrangements used for path planning in robotics. Swiss Federal Institute of Technology, Institute of Theoretical Computer Science, 1999.
11. Samet, H., Neighbor Finding Techniques for Images Represented by Quadtrees, Computer Graphics and Image Processing 18, 37-57, 1982.
12. J.C. Latombe. Robot Motion Planning. Kluwer Academic Publishers, Boston, 1991.
13. B. Chazelle. Approximation and decomposition of shapes. In J. T. Schwartz and C.-K. Yap, editors, Advances in Robotics 1: Algorithmic and Geometric Aspects of Robotics, pp. 145-148. Lawrence Erlbaum Associates, Hillsdale, NJ, 1987.
14. Hart, P., Nilsson, N., & Rafael, B. A formal basis for the heuristic determination of minimum cost paths. IEEE Transactions on Systems Science and Cybernetics, 4, 100–107, 1968.
15. Stentz, Anthony, Optimal and Efficient Path Planning for Partially-Known Environments, Proceedings of the International Conference on Robotics and Automation: 3310–3317, 1994.
16. Daniel K., Nash A., Koenig S., Felner A. Theta*: Any-angle path planning on grids. J. of Artificial Intelligence Research, 2010, vol. 39, pp. 533–579.
17. Koenig, S.; Likhachev, M.; Furcy, D. Lifelong Planning A*. Artificial Intelligence, 2004, vol. 155 (1–2), pp. 93–146.
18. Koenig, S.; Likhachev, M. Fast Replanning for Navigation in Unknown Terrain. Transactions on Robotics, 2005, vol. 21 (3), pp 354–363.
19. Alvarez D., Gomez J.V., Garrido S., Moreno L. 3D robot formations path planning with fast marching square. J. of Intelligent and Robotic Systems. 2015. Vol. 80. No. 3-4. Pp. 507–523.
20. S. Osher and J. A. Sethian, Fronts Propagating with Curvature Dependent Speed: Algorithms Based on Hamilton-Jacobi Formulations, J. of Computational Physics, no. 79, pp. 12–49, 1988.

21. Khatib O. Real-time obstacle avoidance for manipulators and mobile robots // Intern. J. of Robotics Research. 1986. Vol. 5. No. 1. pp. 90–98.
22. Y. Koren and J. Borenstein, “Potential field methods and their inherent limitations for mobile robot navigation,” in Proc. IEEE Conf. Robotics and Automation, Sacramento, CA, Apr. 7–12, 1991, pp. 1398–1404.
23. D. Koditschek, Exact robot navigation by means of potential functions: Some topological consideration, in: Proceedings of IEEE Conference on Robotics and Automation, 1987, pp. 1–6
24. J. Latombe, Robot Motion Planning, 101 Philip Drive, Kluwer Academic Publishers, Assinippi Park, Norwell, MA 02061, 1991
25. N. Franceschini, J. Pichon, C. Blanes, From insect vision to robot vision, Philosophical Transactions of the Royal Society B 337 (1992) 283–294.
26. E. Rimon, D. Koditscheck, Exact robot navigation using artificial potential functions, IEEE Transactions on robotics and automation 8 (5) (1992) 501–518
27. C. Connolly, J. Burns, R. Weiss, Path planning using laplace’s equation, in: Proceedings of IEEE International Conference on Robotics and Automation 3, 1990, pp. 2102–2106.
28. L. Singh, J. Wen, H. Stephanou, Real-time robot motion control with circulatory fields, in: Proceedings of the IEEE International Conference On Robotics and Automation, Minneapolis, Minnesota, 1996, pp. 2737–2742.
29. Borenstein J., Koren Y. Real-time obstacle avoidance for fast mobile robots // IEEE Trans. on Systems, Man, and Cybernetics. 1989. Vol. 19. No. 5. Pp. 1179–1187.
30. Borenstein J., Koren Y. The vector field histogram-fast obstacle avoidance for mobile robots // IEEE Trans. on Robotics and Automation. 1991. Vol. 7. No. 3. Pp. 278–288.
31. Pygame. URL: https://www.pygame.org/news
32. Surprise34/Masters-dissertation: исходный код программы магистерской диссертации. URL:

https://github.com/Surprise34/Masters-dissertation

33. Random — Generate pseudo-random numbers. Python 3.8.3rc1 documentation. URL: https://docs.python.org/3/library/random.html#module-random