АНАЛИЗ АЛГОРИТМОВ ОБХОДА ПРЕПЯТСТВИЙ И ПОИСКА ПУТИ В АПРИОРНО НЕОПРЕДЕЛЕННОЙ СРЕДЕ ДЛЯ МОБИЛЬНОГО УСТРОЙСТВА

Афанасов А.Л.

Афанасов Алексей Леонидович - магистрант,
направление: программная инженерия,
кафедра программной инженерии,
Орловский государственный университет им. И.С. Тургенева, г. Орел

Аннотация: *данная статья посвящена анализу алгоритмов обхода препятствий и поиска пути для мобильного устройства в априорно неопределенной среде. Рассмотрены алгоритмы Жука и навигации по зазорам (GNT), а также методы обхода препятствий с помощью методов клеточной декомпозиции и диаграммы Вороного.*
Ключевые слова: *диаграмма Вороного, мобильное устройство, GNT.*

Для решения задач навигации в условиях непрерывных сред существует два семейства алгоритмов: так называемые алгоритмы Жука и деревья навигации зазора.

При работе с этими алгоритмами используются два основных датчика:

1. Датчик цели указывает текущее расстояние до цели и направление цели, выраженное относительно абсолютного «севера».

2. Локальный датчик видимости обеспечивает точную форму границы на небольшом расстоянии от робота. Робот должен находиться в контакте или почти соприкасаться, чтобы наблюдать за частью границы; в противном случае датчик не предоставляет никакой полезной информации.

Датчик цели по существу кодирует положение робота в полярных координатах (цель - начало координат). Поэтому уникальные (x, y) координаты могут быть назначены на любую позицию, посещенную роботом. Это позволяет ему постепенно отслеживать границы препятствий, которые уже пройдены. Локальный датчик видимости обеспечивает достаточно информации для обеспечения движения на стене; диапазон датчика очень короткий, поэтому робот не может больше узнать о структуре окружающей среды.

Стратегия Жук1 показана на рисунке 1:

1. Двигайтесь к цели, пока не встретится препятствие или цель. Если цель достигнута, остановитесь.

2. Поверните налево и следуйте по всему периметру контактного препятствия. Как только полный периметр посещен, вернитесь к точке, в которой цель была самой близкой, и перейдите к шагу 1.

Худший случай концептуально прост для понимания. Общее расстояние, пройденное роботом, не больше

$d + \frac{3}{2}\sum_{i=1}^{M} p_i$,

где d - расстояние от исходного положения до положения цели, pi - периметр i-го препятствия, а M - количество препятствий. Это означает, что граница каждого препятствия соблюдается не более 3/2 раз. На рисунке 2 показан пример, в котором каждое препятствие пересекается в 3/2 раза. Эта связь основана на том, что робот всегда может вспомнить кратчайший путь вдоль границы к точке, из которой он должен уйти. Это кажется разумным, потому что робот может вывести свое расстояние, пройденное вдоль границы от датчика цели. Если это было невозможно, то 3/2 пришлось бы заменить на 2, потому что робот мог почти пересечь полную границу дважды в худшем случае.

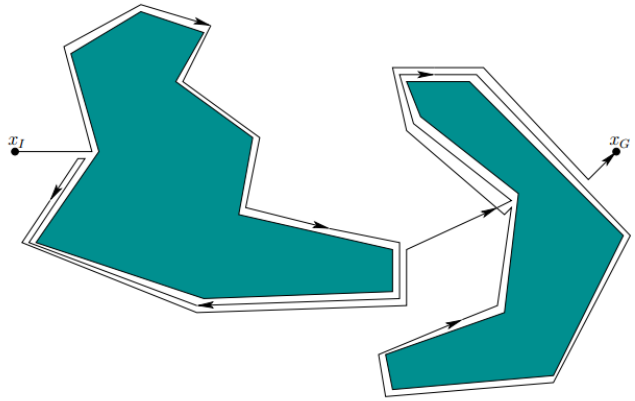


Рис. 1. Иллюстрация стратегии Жук1

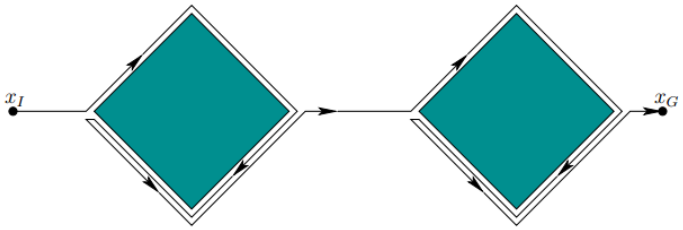


Рис. 2. Плохой пример для Жук1, периметр каждого препятствия пройден 3/2 раза

Стратегия Жук2

Альтернативой Жук1 является стратегия Жук2, которая проиллюстрирована на рисунке 3. Робот всегда пытается двигаться по линии, которая соединяет начальную и конечную позиции. Когда робот находится на этой линии, направление цели будет либо тем же, что и из исходного состояния, либо оно будет отличаться на π радиан (если робот находится на другой стороне цели). Первый шаг такой же, как и для Жук1. На втором этапе робот следует по периметру только до тех пор, пока линия

не будет достигнута, и он не сможет двигаться в направлении к цели. Оттуда он переходит к этапу 1. Однако существует возможность попадания в бесконечные циклы. Поэтому требуется небольшая модификация. Робот помнит расстояние до цели с последней точки, в которой он уходил от границы, и снова отходит от границы, если точка кандидат ближе к цели. Это применяется итеративно до достижения цели или считается невозможным.

Для стратегии Жук2 общее пройденное расстояние не более

$$d + \frac{1}{2}\sum_{i=1}^{M} n_i p_i$$

в которой ni - число раз, когда i-е препятствие пересекает отрезок линии между исходным положением и положением цели. Пример, иллюстрирующий проблему, вызванную пересечениями, показан на рисунке 4.

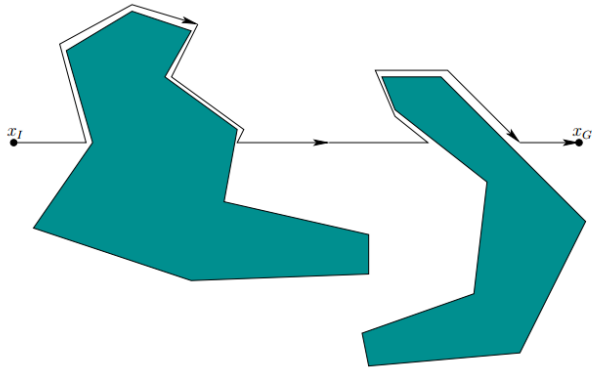


Рис. 3. Иллюстрация стратегии Жук2

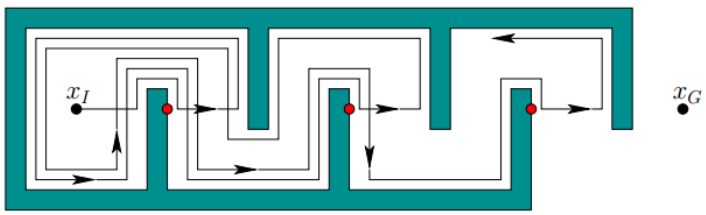


Рис. 4. Плохой пример для стратегии Жук2, где показана лишь часть конечного пути. Точки, из которых робот может покинуть границу, указаны красным

А теперь перейдем к деревьям навигации зазора (GNT) которые представляют собой структуру данных и связанный с ними алгоритм планирования для обеспечения оптимальной навигации в непрерывных средах. В этом разделе предполагается, что робот оснащен датчиком

зазора, который указывает только направления, в которых происходят зазоры, вместо предоставления информации о расстоянии.

В каждый момент времени робот имеет одно действие для каждого зазора, которое видно в датчике зазора. Если действие применяется, то робот перемещается к соответствующему зазору. Это можно применять в течение непрерывного времени, что позволяет роботу «преследовать» конкретный зазор. У робота нет другой информации: он не имеет компаса и не имеет возможности измерять расстояния. Поэтому невозможно построить карту среды, содержащую метрическую информацию.

Предположим, что робот помещен в неизвестную, но просто связанную плоскую среду X. Анализируя, как происходят критические события в датчике зазора, может быть построено древовидное представление, которое указывает, как оптимально перемещаться в окружающей среде, даже если точные измерения не могут быть выполнены. Поскольку датчик зазора не может даже измерять расстояния, может показаться необычным, что робот может двигаться по кратчайшим путям, не получая никакой дистанционной (или метрической) информации.

Внешний вид среды относительно положения робота кодируется как дерево, которое указывает, как зазоры изменяются по мере перемещения робота. Он предоставляет роботу достаточную информацию для перемещения в любую часть среды при движении по кратчайшему пути. Важно понимать, что дерево не соответствует какой-либо статической карте среды. Оно выражает, как среда появляется относительно робота и поэтому может изменяться по мере перемещения робота в окружающей среде.

Корень дерева представляет собой датчик зазора. Для каждого зазора, который в настоящее время появляется в датчике, к корню подключено ребро. Пусть эти ребра называются корневыми ребрами. Каждое корневое ребро соответствует действию, которое может быть применено роботом. Выбирая корневое ребро, робот совершает перемещение по прямой линии к этому промежутку. Таким образом, существует простая модель управления, которая позволяет роботу двигаться точно к определенной точке вдоль границы ∂X.

Пусть V (x) - область видимости, являющаяся множеством всех точек в X, которые видны из x. Пусть X \ V (x) будем называть теневой областью, которая является множеством всех точек, не видимых из x. Пусть каждая связанная компонента теневой области называется теневой компонентой. Каждый зазор в датчике зазора соответствует сегменту линии в X, который касается ∂X в двух местах.

На рисунке 5 показано применение действия преследования зазора, которое перемещает робота прямо в его направлении до контакта с границей. Как только это произойдет, становится видимой новая часть среды.

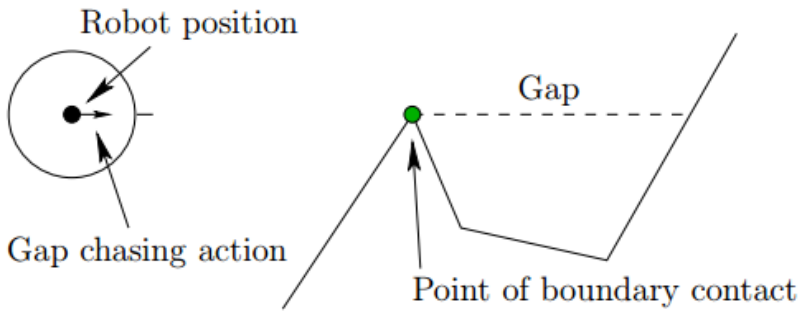


Рис. 5. Действие преследования зазора

Каждый из этих сегментов образует границу между областью видимости и теневым компонентом. Если робот хотел бы перейти к этому теневому компоненту, самый короткий путь - это перейти непосредственно к зазору. При перемещении к зазору робот в конечном итоге достигает ∂X, и в этот момент необходимо выбрать новое действие.

События критического зазора.

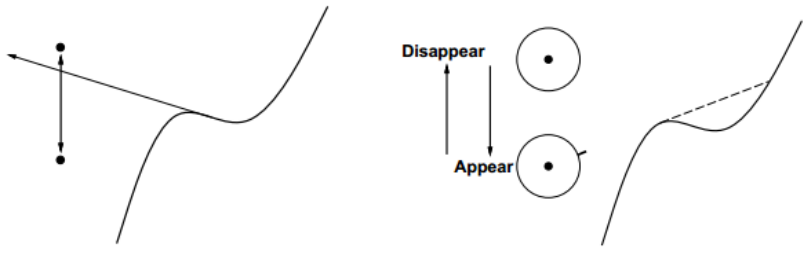
По мере движения робота в датчике зазора могут возникать несколько важных событий:

1. Исчезновение: зазор исчезает, потому что робот пересекает луч перегиба, как показано на рисунке 6(a). Это означает, что теперь отображается предыдущий теневой компонент.

2. Появляется: появляется зазор, потому что робот пересекает луч изгиба в противоположном направлении, что можно увидеть на рисунке 6(b). Это означает, что существует новый теневой компонент, который представляет собой недавно скрытую часть среды.

3. Разделение: зазор разделяется на два зазора, потому что робот пересекает луч-битангент, как показано на рисунке 7(a). Это означает, что один теневой компонент разбивается на две составляющие тени.

4. Слияние: два зазора сливаются в один, потому что робот пересекает луч-битангент в противоположном направлении на рисунке 7(b). В этом случае две теневые компоненты сливаются в одну.



(a) (b)

Рис. 6 (a) - Робот пересекает луч, который простирается от точки перегиба. (b) - Зазор появляется или исчезает из датчика зазора в зависимости от направления

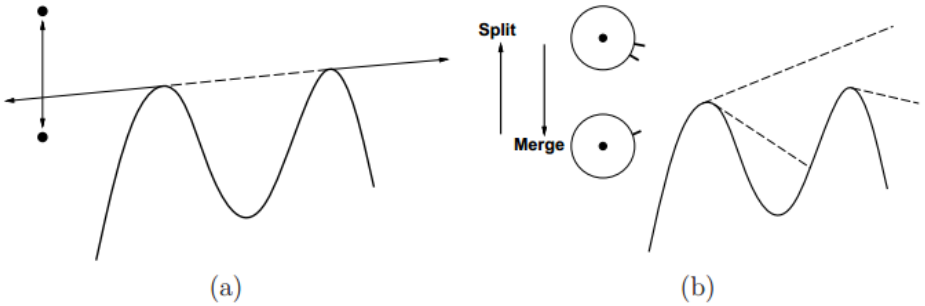


Рис. 7. (a) - Робот пересекает луч, который простирается от битангента. (b) - Зазор или слияние в зависимости от направления

Это полный список возможных событий, в предположении общего положения, исключающих среды, которые вызывают вырождения, такие как три зазора, которые сливаются в один или вид зазора, где разрываются два других зазора.

Поскольку каждое из этих событий зазора происходит, оно должно отражаться в дереве. Если зазор исчезает, как показано на рисунке 8, то соответствующие ребра и вершина просто удаляются. Если происходит событие слияния, то промежуточная вершина вставляется, как показано на рисунке 9. Это указывает на то, что если этот зазор будет преследован, он разделится на два первоначальных пробела. Если происходит разделение, как показано на рисунке 10, то промежуточная вершина удаляется. Появление зазора является важным случаем, который генерирует примитивную вершину в дереве, как показано на рисунке 11.

Обратите внимание, что примитивная вершина никогда не может разделиться, потому что преследование ее приведет к ее исчезновению.

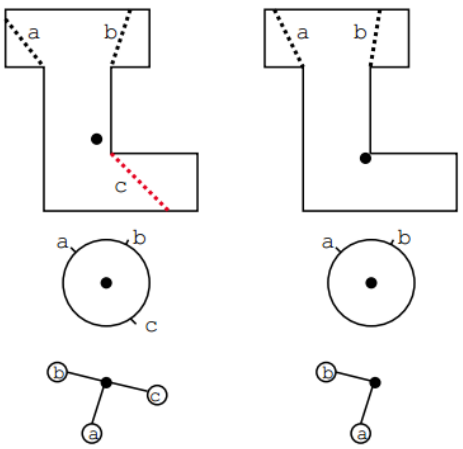


Рис. 8. Если зазор исчезает, он просто удаляется из GNT

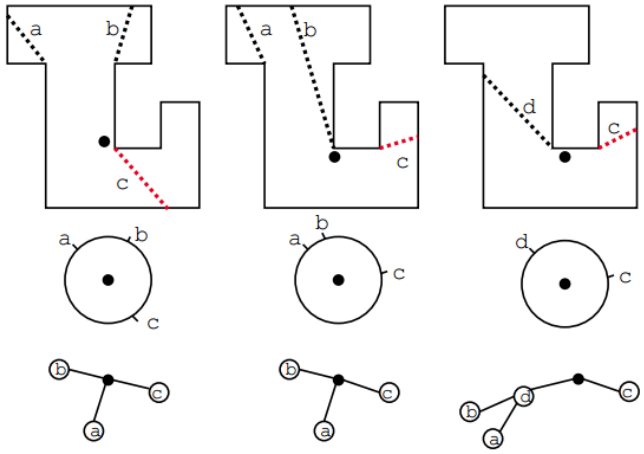


Рис. 9. Если сливаются два зазора, в дерево вставляется промежуточная вершина

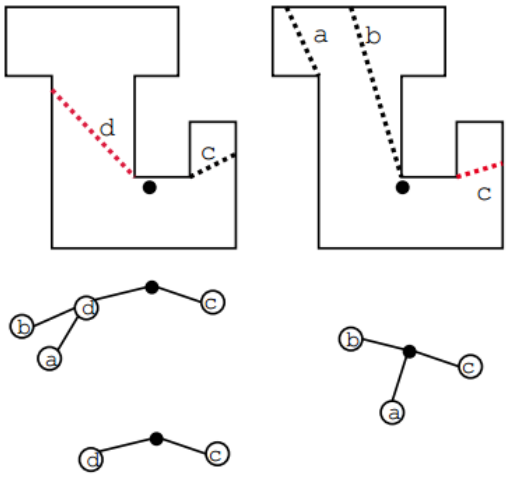


Рис. 10. Если разделяются два зазора, из дерева удаляется промежуточная вершина

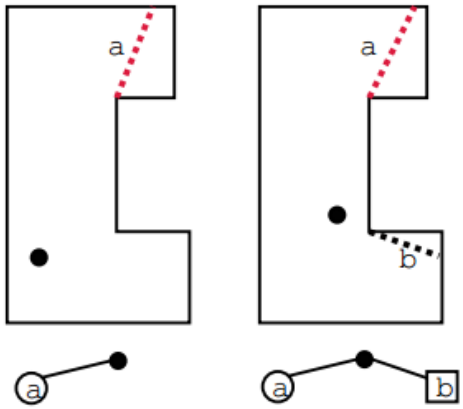


Рис. 11. Появление зазора приводит к примитивной вершине, которая обозначается квадратом

Кроме задачи поиска кратчайшего пути существует еще одна важная задача – обход препятствий и построение пути в конфигурационном пространстве. Под данным пространством понимается гипотетическое (абстрактное) многомерное пространство, отражающее все множество конфигураций какой-нибудь сложной системы. Под конфигурациями в данном случае можно понимать взаимное положение всех частей системы.

В данном случае робот сокращается до точки в свободном пространстве Cfree (Рис. 12), что позволяет облегчить поиск пути и применить классические методы декомпозиции.

Рис. 12. Абстракция конфигурационного пространства (Cfree)

Всего можно выделить три подобных метода:

— клеточная декомпозиция;

— карты кратчайших путей;

— карты максимальных зазоров.

Пример для вертикальной клеточной декомпозиции заключается в разбиении среды на выпуклые части, затем построение карты, соединяющей разбитые куски, установка точек X1 XG и подключение к центрам клеток. Последним шагом является поиск пути по построенной на втором шаге карте (Рис. 13).

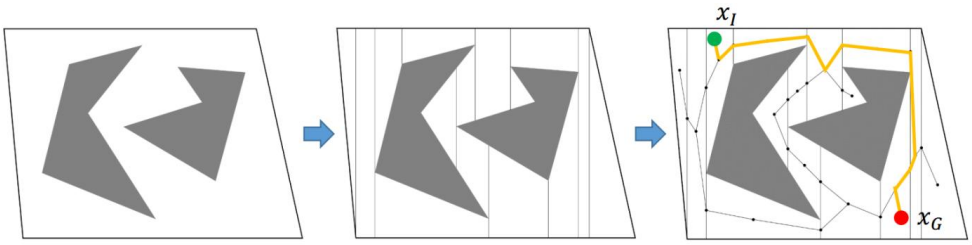


Рис. 13. Пример клеточной декомпозиции

Вертикальная клеточная декомпозиция проста, но является неоптимальной, она не находит кратчайшего пути из X1 в XG. Данную проблему решают карты кратчайших путей (также известные как редуцированный граф видимости).

Этап построения

Рефлексная вершина — это та, где угол через вершину в окружении больше, чем π (Рис. 14).

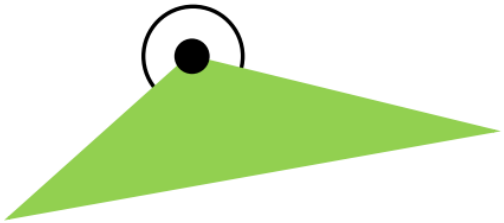


Рис. 14. Рефлексная вершина

Зная рефлексные вершины, добавим их все в карту по следующим принципам:

— если эти вершины являются последовательными для одного препятствия;

— если две вершины дают битангент;

— несквозные вершины не добавляются.

Поиск кратчайшего пути (Рис. 15):

— Добавьте X1 и XG на карту;

— Соедините X1 с видимой вершиной дорожной карты, то же самое для XG;

— Выполните поиск кратчайшего пути по подключенной дорожной карте, содержащей X1 и XG.

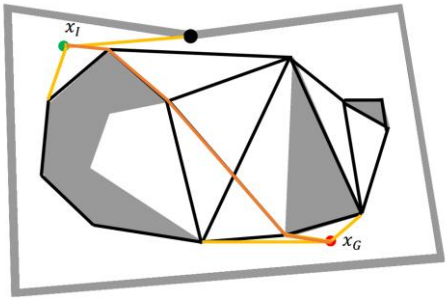


Рис. 15. Пример построения пути

В карте максимальных зазоров требуется, чтобы роботы избегали границ препятствий, так как это небезопасно. Карта максимальных зазоров решает проблему, основываясь на идее диаграммы Вороного (Рис. 16).

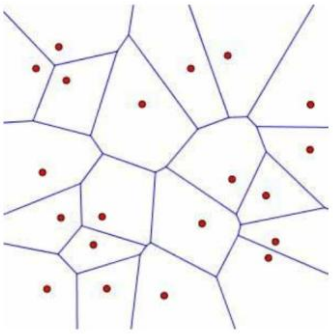


Рис. 16. В диаграмме Вороного линии находятся на максимально возможном удалении от точек

При построении пути помимо взаимодействий вершина-вершина используются взаимодействия с ребрами (Рисунок 17), а на рисунке 18 можно увидеть пример построения пути.

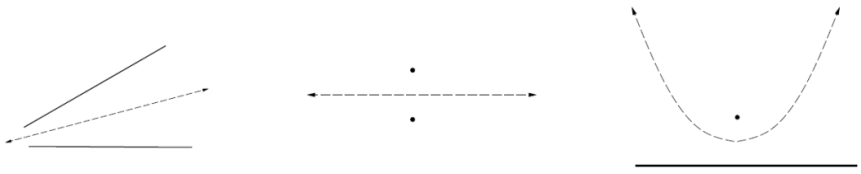


Рис. 17. Варианты взаимодействия

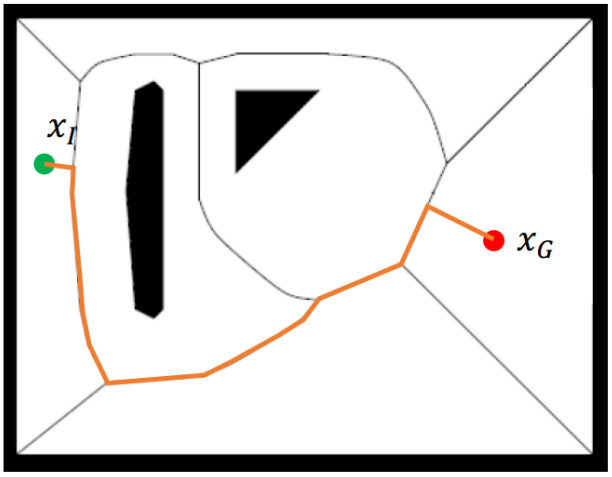


Рис. 18. Пример построения

В данной работе описаны основные методы обхода препятствий и планирования пути в среде с препятствиями при отсутствии априорной информации об окружающей среде и выделены критические точки рассмотренных алгоритмов.

Список литературы

1. Robotic Motion Planning: Sample-Based Motion Planning [Электронный ресурс]. Режим доступа: https://www.cs.cmu.edu/~motionplanning/lecture/Chap7-Prob-Planning_howie.pdf/ (дата обращения: 27.05.2019).
2. Introduction to Computational Robotics. [Электронный ресурс]. Режим доступа: https://arc.cs.rutgers.edu/courses/f17/lecture.19.pdf/ (дата обращения: 12.06.2019).
3. *LaValle Steven.* Planning algorithms. [Электронный ресурс]. Режим доступа: http://planning.cs.uiuc.edu/bookbig.pdf/ (дата обращения: 16.06.2019).