Лашков А.А., Зернов М.М. Применение навигационного графа для решения задачи поиска маршрута габаритного объекта по цифровой карте местности // Международный журнал информационных технологий и энергоэффективности. – 2022. – Т. 7 № 1(23) с. 3-16

Международный журнал информационных технологий и энергоэффективности

Сайт журнала:
http://www.openaccessscience.ru/index.php/ijcse/

УДК 681.3.019

ПРИМЕНЕНИЕ НАВИГАЦИОННОГО ГРАФА ДЛЯ РЕШЕНИЯ ЗАДАЧИ ПОИСКА МАРШРУТА ГАБАРИТНОГО ОБЪЕКТА ПО ЦИФРОВОЙ КАРТЕ МЕСТНОСТИ

[1]Лашков А.А., [2]Зернов М.М.
[1,2]Филиал ФГБОУ ВО «Национальный исследовательский университет «МЭИ» в г. Смоленске, Россия, (214013, г. Смоленск, Энергетический проезд, 1), e-mail: [1]lahkandr010@mail.ru, [2]zmmioml@yandex.ru

Данная статья посвящена рассмотрению возможных путей решения задачи поиска маршрута площадного объекта с использованием навигационного графа. Дано описание навигационного графа. Выделены этапы и особенности реализации построения навигационного графа. Описаны некоторые методы ускорения построения навигационного графа. Рассмотрены методы поиска кратчайшего пути на графах.

Ключевые слова: навигационный граф, КД-дерево, скелет, диаграмма Вороного.

THE USE OF A NAVIGATION GRAPH TO SOLVE THE PROBLEM OF FINDING THE ROUTE OF A DIMENSIONAL OBJECT ON A DIGITAL TERRAIN MAP

[1]Lashkov A.A., [2]Zernov M.M.
Smolensk Branch of the National Research University "Moscow Power Engineering Institute", Smolensk, Russia (214013, Smolensk, Energetichesky proezd, 1), e-mail: [1]lahkandr010@mail.ru, [2]zmmioml@yandex.ru

This article is devoted to the consideration of possible ways to solve the problem of finding the route of an area object using a navigation graph. A description of the navigation graph is given. The stages and features of the implementation of the construction of the navigation graph are highlighted. Some methods of accelerating the construction of a navigation graph are described. Methods of finding the shortest path on graphs are considered.

Keywords: navigation graph, KD-tree, skeleton, Voronoi diagram.

Введение

Поиск маршрута с учётом пространственных размеров перемещаемого объекта – это нахождение последовательности допустимых положений объекта при перемещении его в пространстве от источника к месту назначения. На практике поиск маршрута даже в простых случаях с относительно небольшим количеством препятствий становится трудноразрешимой задачей, если требуется учесть габариты объекта.

Имеется цифровая карта местности (ЦКМ) [1,2,3], описанная в двухмерном пространстве, на которой расположены препятствия (горы, дома, реки, бездорожье и т.д.) и сам перемещаемый объект. Задача заключается в определении кратчайшего маршрута между двумя заданными точками, по которому может пройти объект. При этом необходимо

учитывать проходимость местности, т.к. объект имеет габариты, то он может просто-напросто не пролезть между окружающими его препятствиями или не пройти по бездорожью. Нахождение маршрута для точечного объекта является уже давно известной и решаемой задачей. Тогда как поиск маршрута объекта с учетом его габаритов до сих пор практически не рассматривался. Анализ существующих методов построения маршрута с использованием карты местности позволяет утверждать, что необходимый математический аппарат для нахождения маршрута площадного объекта (цепочка точек соединенные отрезками) уже имеется и необходимо разработать способ на основе известных методов и алгоритмов. В качестве объекта может выступать техника, группа людей, любой крупногабаритный объект, линейные размеры, которого сопоставимы с расстояниями между местными объектами.

Данная статья посвящена рассмотрению возможных путей решения задачи поиска маршрута площадного объекта с использованием графов, совокупности алгоритмов для построения наиболее подходящего вида графов – навигационного графа и алгоритмов поиска пути в графе.

Навигационный граф

Как правило задачи поиска или нахождения маршрута решаются на основе методов на графах и в основном для точечных объектов, например, в геоинформационных системах и компьютерных играх. В первом примере идет речь о графе дорог [4,5], а во втором используется граф точек изменения маршрута [6,7].

Подход на основе графа дорог не учитывает возможности движения по бездорожью и требует дополнительной адаптации на основе анализа проходимости ЦКМ. Это сближает с подходом принятом в компьютерных играх, где на основе анализа игровой карты строится граф точек изменения маршрута (waypoints).

Тем не менее и граф дорого и граф точек перемены маршрута обладают общим недостатком – отсутствием информации о размерах проходимых областей для их соотнесения с размерами(шириной), перемещаемого объекта. Требуемыми свойствами обладает другая структура данных – навигационный граф [8,9].

Навигационный граф является результатом клеточной декомпозиции рассматриваемой среды (ЦКМ). Декомпозиция не является точной, поскольку вычисление основано на дискретном представлении препятствий. В данный граф входят следующие элементы:

- вершины(ячейки) представляют собой окружности (в 3-мерном случае - цилиндры), внутри которых находится свободное от препятствий пространство;
- ребра, соединяющие вершины, являются отрезками, образующимися точками пересечения вершин(окружностей).

Основное свойство навигационного графа состоит в том, что любая точка, принадлежащая данной ячейке, может достигать любой точки, принадлежащей соседней ячейке, проходя мимо любой точки их пересечения (в зависимости от условий). Таким образом нахождение маршрута площадного объекта сводится к задаче поиска пути в навигационном графе. Для этого его сперва необходимо построить.

Процесс построения навигационного графа имеет следующую структуру:

1. Формирование карты проходимости на основе ЦКМ, выделение свободных областей и препятствий.

2. Формирование карты зазоров или дистанционной карты. Для каждой свободной точки находится расстояние до ближайшего препятствия.

3. Для того чтобы компактно расположить вершины навигационного графа по свободной области нужно чтобы вершины занимали как можно больше этой площади, значит, имели максимально возможный радиус. Для выбора таких окружностей может помочь скелет свободной области ЦКМ.

4. Выбор вершин для графа, на основе занимаемых окружностями площадей, и их соединение.

Вышеперечисленные этапы могут приводить к большим вычислительным затратам, чтобы ускорить процесс расчета можно поделить ЦКМ на прямоугольные участки меньшей площади, обеспечивающие быстрый поиск препятствий, например, организованные в виде KD-дерева.

Предварительная обработка цифровой карты местности

В настоящее время цифровое картографирование широко распространено, и во всех отраслях создается все больше геоданных, которые и заносятся на цифровые карты местности. ЦКМ – представление картографических объектов в форме, позволяющей сохранять, обрабатывать и выводить значения их атрибутов компьютерной системой. ЦКМ несет информацию, которую можно анализировать обрабатывать и преобразовывать в формат, который необходим для решения специализированных задач [3].

Нам необходимо преобразовать ЦКМ в форму удобную для построения навигационного графа, а для этого нужно выделить область, занимаемую препятствиями (бездорожьем) и свободную область. В результате решения, данной задачи формируется матрицы проходимости участков карты, определённых в соответствии с регулярной сеткой. Каждая ячейка матрицы будет содержать коэффициент проходимости соответствующего участка (квадрата), изменяемый от 0(свободен) до 1(непроходим или является точкой препятствия). На основе матрицы проходимости, учитывая выбор граничного значения коэффициента проходимости для нахождения точек препятствий (например, выбрать все точки как препятствия, значения коэффициента в матрицы проходимости которых больше 0.5), строят двухмерную карту.

Двухмерная карта представляет собой растровую решетку. Растровая решетка называется прямоугольная решетка Z^2, состоящая из точек евклидовой плоскости с целочисленными координатами. Решетка образуется из точек пресечения двух взаимно перпендикулярных семейств параллельных прямых. Опишем понятие соседства точек:

- непосредственные соседи – точки, евклидово расстояние между которыми равно 1;
- косвенные соседи - точки, расстояние между которыми равно $\sqrt{2}$.

Соответственно можно выделить на основе соседства в растровой решетке 4-смежность и 8-смежность. 4-смежность – смежность, при которой соседними к данной точке являются только непосредственные соседи, а для 8-смежности еще и косвенные соседи. На основе соседства можно сформировать граф смежности, вершинами которого являются точки растровой решетки, а ребрами соединяются все пары соседних вершин. Множество точек может быть связным, если граф смежности является связным, иначе разделенным.

Карта на растровой решетке задается точками решетки двух цветов. Множеством черных точек растровой решетки заданы препятствия, множеством белых – свободные точки. Препятствия являются дискретными фигурами - максимальное связное множество черных точек в растровой решетку. Граничные точки препятствий определяется по соседству 4-х, если какая-либо из четырех соседних точек является свободной, то данная точка является границей.

KD-дерево

Как уже было сказано ранее для упрощения и ускорения построения навигационного графа можно воспользоваться KD-деревом [10,11].

KD-дерево (K-мерное дерево) - структура данных, позволяющая разбить K-мерное пространство делением этого пространства на меньшие участки. Само деление происходит (K-1)-мерными пространствами, например, для плоскости (K=2) прямой.

Такое разбиение позволит сузить диапазон поиска в K-мерном пространстве. Например, поиск ближайшей границы к точке.

Объектами для деления являются точки препятствий, они помещаются в большой ограничивающий прямоугольник (ограничивающим назовем такой прямоугольник, который описывает исходное множество объектов или сам объект, если строится ограничивающий прямоугольник лишь для него), который равен размеру двухмерной карты. Затем данный прямоугольник делится прямой перпендикулярной к одной из сторон, тем самым мы получаем два новых узла дерева (ограничивающих прямоугольника). Новые узлы делятся аналогично и т.д. Деление продолжается по достижению нужной глубины дерева, либо по специальному критерию, либо по количеству объектов в узле.

Опишем некоторые из способов деления узла:

1) В качестве секущей выбрать середину наибольшей стороны.

2) В качестве секущей выбрать такую, что количество объектов слева и справа примерно будет равной.

3) Использовать чередование сторон, т.е. сначала для деления брать середину, допустим, по оси X, далее в новых узлах брать середину по оси Y и т.д.

4) Использовать SAH (Surface Area Heuristic) – функцию оценки разделения ограничивающего прямоугольника [12].

Значение SAH функции вычисляется следующим образом:

$$f(x) = C_t + C_i * \frac{SA_L * N_L + SA_R * N_R}{SA_{parent}},$$

где C_t и C_i – коэффициенты стоимости;

SA_L и SA_R – площади поверхности левого и правого узлов, образуемые в результате деления родительского узла, данным отрезком;

N_L и N_R – количество элементов в левом и правых узлах, после возможного деления родительского узла, данным отрезком;

SA_{parent} – площадь поверхности родительского узла.

Критерий остановки деления узла:

$f_0 = C_i * N$ – SAH родительского узла;

$f_0 < f(x)$ – если выполнено данное неравенство, прекращаем деление.

Т.е. если стоимость прослеживания дочерних узлов больше стоимости прослеживания родительского узла, то деление прекращается.

Дерево строится от корня. В каждом узле дерева хранятся указатели на левое и правое поддеревья, если таковых у узла нет, то он считается листовым (листом). Каждый узел хранит ограничивающий прямоугольник, который описывает объекты данного узла. В листовых и только в листовых узлах хранятся индексы тех объектов, которые входят в данный узел.

Пример структура KD-дерева, описанной на языке C++ представлен ниже [11]:

```
constexpr int N=10; // количество пространств ключей

struct Item {    // структура элемента
  int key[N];    // массив ключей определяющих элемент
  char *info;    // информация элемента
};

struct Node {    // структура узла дерева
  Item i;        // элемент
  Node *left;    // левое  поддерево
  Node *right;   // правое поддерево
}
```

Структура дерева может меняться в зависимости от деталей реализации алгоритма. Например, в узле может содержаться не один элемент, а массив, что повышает эффективность поиска.

Скелетное представление фигуры

Для анализа фигуры описание границ не всегда является информативной и требуется дополнительная форма представления – скелет, который обычно подчеркивает метрические и топологические свойства формы. Скелет является равноудалённой от границ тонкой версией фигуры. Иначе можно воспользоваться определением, основанным на понятии максимального пустого круга [13].

Пустой круг — это окружность, находящаяся в фигуре целиком включая внутренние точки. Пустой круг, который не содержится в других пустых кругах, является максимальным. Множество центров всех максимальных кругов фигуры называется скелетом. Для компактного расположения вершин(окружностей) навигационного графа как раз и нужны такие окружности как можно большего радиуса. Рассмотрим построение скелета на основе дистанционной карты и на основе диаграммы Вороного.

На дискретной плоскости нахождение точек скелета может быть выполнена на основе дистанционной карты. Дистанционная карта – это карта, в которой каждой точки дискретной фигуры ставит в соответствие расстояние до ближайшей границы. Для определения точек скелета используется некоторое локальное правило, если значение расстояния в окрестности

точки по любой из диагоналей, вертикали или горизонтали больше в данной точке, то она является точкой скелета. Такое правило может привести к разрывам и пропуску точек в скелете, поэтому необходимо включать дополнительную постобработку, которая заключается в склеивании отдельный частей скелета. Данный алгоритм относится к дискретным алгоритмам построения скелета.

Дадим описание диаграмме Вороного для того, чтобы было понятней построение скелета на ее основе. Диаграмма Вороного представляет собой разбиение плоскости на области, близкие к каждому из заданного набора объектов. В простейшем случае эти объекты представляют собой конечное число точек на плоскости (называемых начальными точками, узлами или сайтами) [14].

Пусть P – заданное на плоскости конечное множество точек, называемые сайтами. Для каждого сайта $p \in P$ существует соответствующая область, называемая ячейкой Вороного, состоящая из всех точек плоскости, расположенных ближе к этому сайту, чем к любому другому.

Рассмотрим множество из двух сайтов $P = \{p, q\}$, соединим эти точки и проведем срединный перпендикуляр полученного отрезка $[p, q]$, который называется бисектором. Получили две полуплоскости H_{pq} и H_{qp}, разделенных бисектором сайтов p и q, эти полуплоскости являются ячейками диаграммы Вороного (рисунок 1).

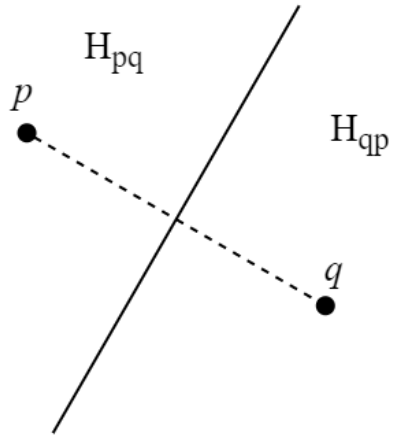

Рисунок 1 – Деление на полуплоскости

Если взять большее множество сайтов $P = \{p, q_1, \dots, q_n\}$, то ячейкой Вороного V_p для сайта p будет являться пересечением всех полуплоскостей H_{pq_i} (рисунок 2).

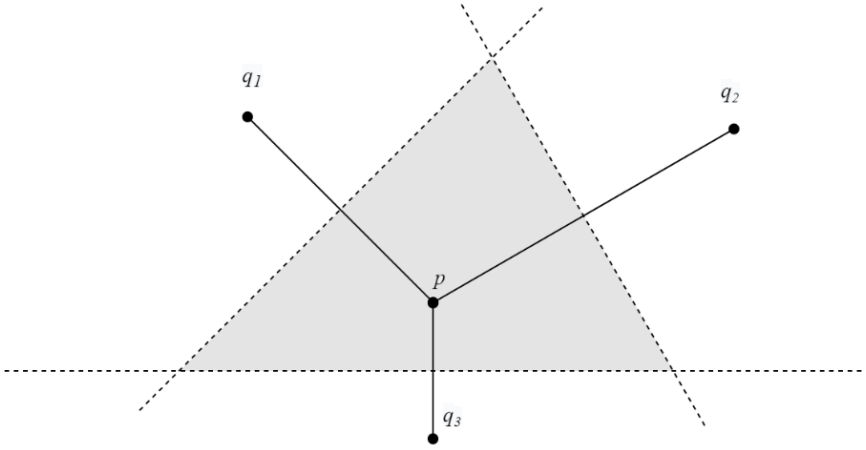

Рисунок 2 – Ячейка Вороного

Совокупность ячеек всех сайтов образуют диаграмму Вороного (рисунок 3). Алгоритм для нахождения каждой ячейки Вороного таким образом может быть реализован с вычислительной сложностью $O(n^4)$[13].

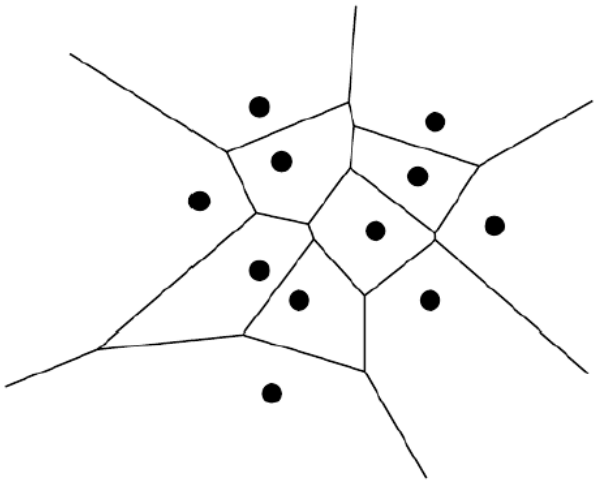

Рисунок 3 – Диаграмма Вороного

Если в качестве сайтов взять точки границы препятствий, то ребра диаграммы Вороного можно использовать в качестве элементов скелета двухмерной карты, т.к. данные ребра равноудалены от блажащих сайтов, но вместе с нужными ребрами диаграмма будет содержать еще «шумовые» ребра, которые не подходят для составления скелета (рисунок 4,а). Поэтому необходимо включить процесс исключения лишних ребер(«стрижка») диаграммы Вороного (рисунок 4,б).

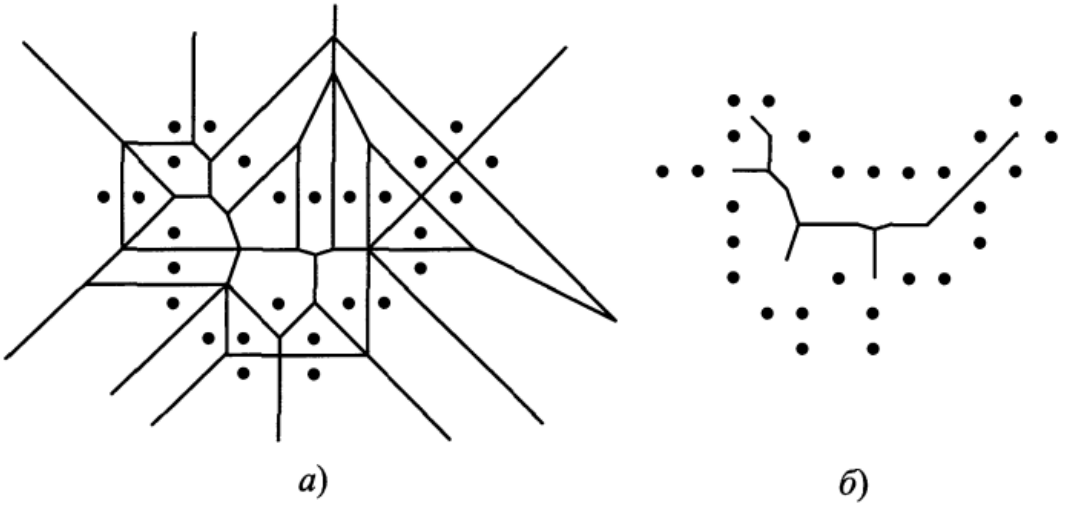

Рисунок 4 – Диаграмма Вороного граничных точек и соответствующий ей скелет без лишних ребер

Полученный после «стрижки» [13,15,16] скелет является непрерывным скелетом, в отличии от скелета, полученного с помощью, например, дистанционной карты.

Алгоритм Форчуна

Построение диаграммы Вороного множества точек на плоскости является сложным и трудоемким процессом. Для построение данной диаграммы можно использовать алгоритм Форчуна, являющимся одним из быстрых алгоритмов, он генерирует диаграмму Вороного из множества точек на плоскости за время $O(n * \log(n))$ [17]. Этот алгоритм основан на заметающей прямой, которая двигается слева направо. Заметающая прямая предоставляет собой вспомогательный объект, являющийся вертикальной прямой линией.

Основным недостатком является то, что вычисления выполняются с плавающей запятой вместо целых чисел, что открывает путь к ошибкам точности. Однако алгоритмы, использующие только целые числа, имеют риск переполнения целых чисел даже при достаточно малых ограничениях.

Напомним, что множество точек, равноудаленных от точки и прямой(директрисы), задается параболой. Теперь, если мы построим параболу для каждого сайта, где сайт является фокусом, а заметающая прямая является директрисой, все, что находится справа от всех парабол, будет неопределенной областью, а слева определённой. Граница областей определенности и неопределенности называется береговой линией, состоящей из параболических дуг. Каждая парабола будут меняться в зависимости от положения заметающей прямой относительно сайта — чем дальше она уходит от сайта вправо, тем больше расширяется парабола, однако в самом начале она вообще является лучом.

Двигая заметающую прямую слева направо, сохраняя структуру береговой линии, могут происходить следующие два типа событий, которые изменяют структуру береговой линии: событие точки и событие вершины.

В событии точки заметающая прямая обнаруживает сайт и вставляет ее параболическую дугу в береговую линию.

В событии вершины параболическая дуга исчезает с береговой линии в точке, соответствующей вершине диаграммы Вороного, просто удаляется дуга, длина которой уменьшается до нуля. Каждый раз, когда изменяется береговая линия, обновляются соответствующие точки для события вершины.

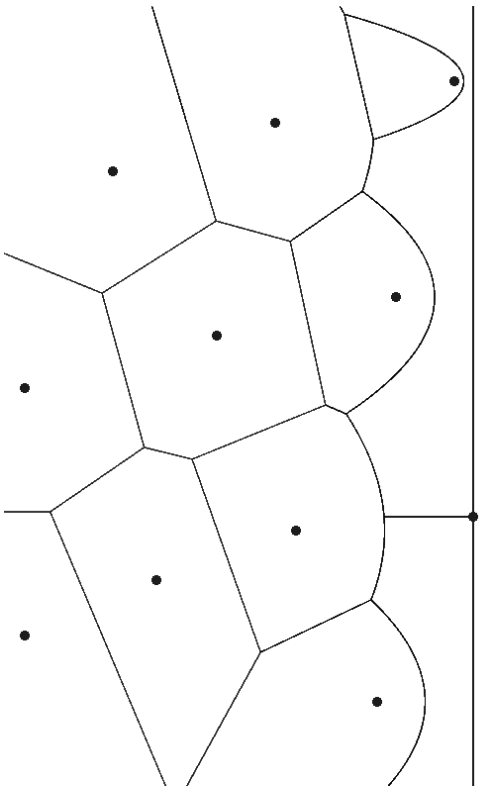

Рисунок 5 – Алгоритм Форчуна

Алгоритм Форчуна является не единственным алгоритмом построения с такой асимптотикой, но он достаточно понятен и не очень сложен в реализации.

Если препятствия на карте представлены многоугольниками, то можно воспользоваться алгоритмом «сегментных границ». Т.е. представить границу многоугольника в виде множества элементов – ее вершин и сторон. Стороной является отрезок границы с инцидентными ему вершинами. Сайтами в данном алгоритме являются стороны(сегменты) и вершины многоугольников. Алгоритм сегментный границ описан в [18].

Построение навигационного графа

Вершина графа создается из точки скелета, используя ее в качестве центра и соответствующий ей зазор(расстояние до ближайшей границы препятствия) в качестве радиуса. Точка скелета с максимальным зазором выбирается для создания вершины графа, и все точки скелета, покрытые этой вершиной, исключаются из выбора. Затем процесс повторяется до тех пор, пока не останется больше точек скелета для выбора. После нахождения всех вершин соединяем ребрами те из них, которые пересекаются. Пример навигационного графа изображен на рисунке 6.

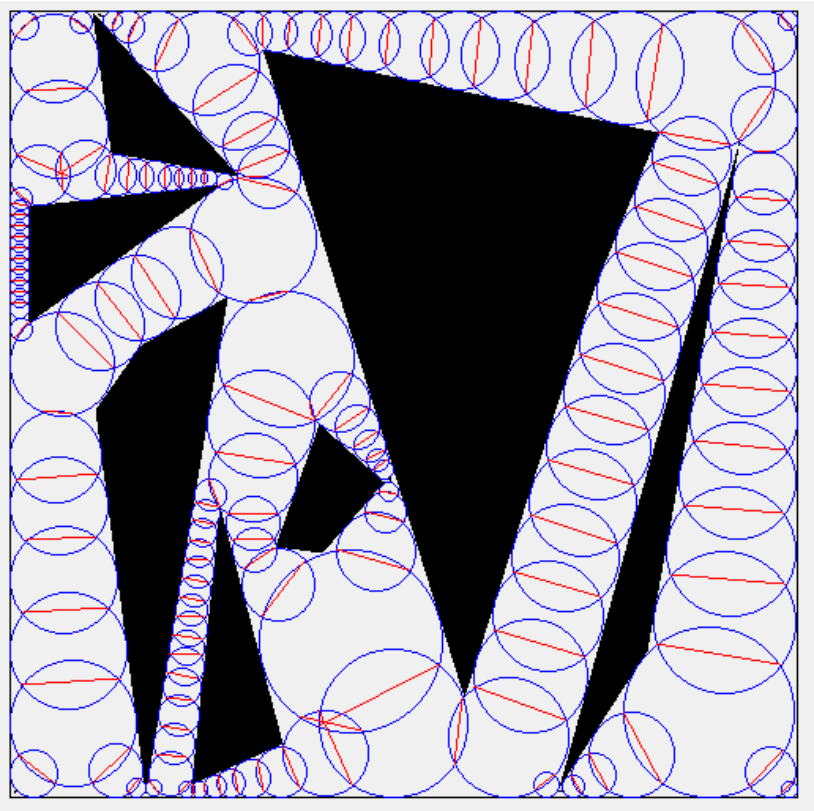

Рисунок 6 – Навигационный граф

Взвешивание графа

Для алгоритмов поиска на графах необходимо взвесить граф. Вес вершины – радиус. Вес ребра графа выбран векторным (две компоненты):

1) расстояние между центрами соседних вершин (оценка времени прохождения между вершинами);

2) ширина коридора (расчет может быть произведен по-разному, например, минимум из диаметров соседних вершин или же ширина области пересечения окружностей).

Исходя из заданной ширины перемещаемого объекта и весов ребер навигационного графа происходит исключение ребер, по которым объект не может перемещаться, и применяется один из алгоритмов поиска кратчайшего пути на графе.

Поиск на графе

Маршрут на навигационном графе может быть найден с помощью любого алгоритма поиска на графе, учитывающего веса ребер. Ниже рассмотрены примеры подходящих алгоритмов поиска.

Алгоритм Дейкстры решает задачу нахождения кратчайшего пути от исходной вершины на графе (источника) до пункта назначения [19]. Оказывается, что ко всем точкам графа можно найти кратчайшие пути от заданного источника за одно и то же время, поэтому эту проблему

иногда называют проблемой кратчайших путей с одним источником. Алгоритм работает только для графов без рёбер отрицательного веса.

Алгоритм Дейкстры включает следующие шаги:

1) Устанавливаются расстояние между всеми вершинами равное бесконечности, за исключением начальной вершины, в которой устанавливается расстояние равное 0.

2) Помечаются все вершины, включая начальную, в качестве не посещаемого узла.

3) Выбор из не посещённых вершин с наименьшим текущим весом (меткой вершины) в качестве текущей вершины «C».

4) Назовем «соседями» вершины, в которые ведут ребра из текущей вершины «C». Для каждого соседа «N» текущей вершины «C», кроме отмеченных как посещённые, добавляем метку равную сумме значения метки в текущей вершине «C» и веса ребра «C»-«N», если она меньше текущего значения в вершине «N».

5) Отмечаем текущий узел «C» как посещенный.

6) Необходимо повторить описанный выше шаги с шага 3 до тех пор, пока не будет посещен пункт назначения.

Алгоритм завершается после достижения всех вершин графа. В результате все посещенные вершины будут иметь метку, значение которой равно кратчайшему расстоянию от вершины источника до данный вершины. Могут быть ситуации, когда в графе есть вершины, до которых нельзя добраться, тогда алгоритм можно завершить досрочно.

Алгоритм A * похож на алгоритм Дейкстры, и единственное отличие состоит в том, что A * пытается найти «лучший» путь, используя эвристическую функцию «расстояние + стоимость» (обозначим как f(x)), которая отдает приоритет узлам, которые должны быть «лучше» других, в то время как алгоритм Дейкстры просто исследует все возможные пути. Функция f(x) — сумма двух других: функции стоимости достижения рассматриваемой вершины (x) из вершины источника (обозначим как g(x)), и функции эвристической оценки расстояния от рассматриваемой вершины к конечной (обозначим как h(x)) [20].

Обозначим веса вершин:

- G - стоимость передвижения из начальной вершины к данной вершине, следуя найденному пути к этой клетке;
- F – стоимость равная сумму G и функции h(x) для данной вершины.

Шаги алгоритма A*:

1) Формирование «открытого» списка (в начале работы алгоритма содержит вершину источник), в котором находятся вершины, которые необходимо просмотреть, и «закрытого» списка, который содержит просмотренный вершины.

2) Выбрать в качестве текущей вершины «C» вершину с наименьшим значением F и поместить в закрытый список.

3) Нахождение всех соседних вершин «N», который не входят в закрытый список.

4) Расчет для каждого соседа «N» временного G_TEMP равное сумме веса G текущей вершины «C» и веса ребра «C»-«N». Если «N» находится в открытом списке и его текущее G больше временного G_TEMP, или «N» не находится в открытом списке, то соответственно обновляем в отрытом списке или добавляем в отрытый список вершину «N», где G = G_TEMP и рассчитываем F, еще указываем для «N» вершину, из которой пришли(«C»).

5) Необходимо повторить описанный выше шаги с шага 2 до тех пор, пока не будет посещен пункт назначения.

Заключение

В статье были рассмотрены пути поиска маршрута площадного объекта с использованием навигационного графа. Рассмотрен процесс построения навигационного графа и выделены основные его этапы. Описаны вспомогательные методы ускорения построения навигационного графа (построение скелета и КД-дерева), а также приведены примеры подходящих алгоритмов поиска кратчайшего пути для данного графа.

Процесс построения навигационного графа, основанный на клеточной декомпозиции, является неточным, т.к. навигационный граф не захватывает все свободное от препятствий пространство, а вычисление графа использует дискретное представление окружающей среды. Поэтому для захвата узких проходов может потребоваться высокоточная сетка, что приводит к длительным расчетам.

В дальнейшем предстоит определить наиболее рациональный способ построения скелета и выбор вершин навигационного графа, как с точки зрения минимизации вычислительных затрат, так и с точки зрения максимального покрытия свободного пространства. Предстоит выбрать наиболее рациональный способ усечения навигационного графа в соответствии с габаритами объекта для дальнейшего поиска маршрута алгоритмами Дейкстры или A*.

Список литературы

1. ГОСТ 28441—99. Картография. Термины и определения / Внесён Госстандартом России. — Взамен ГОСТ 28441-90. — М.: ИПК Издательство стандартов, 2000. — С. 2. — 15 с. — (ГОСТ).
2. Цветков В. Цифровые карты и цифровые модели //Электронная версия размещается на сайте www. rae. ru. – 2000. – С. 348.
3. Абламейко С. В., Крючков А. Н. Информационные технологии создания и обновления цифровых и электронных карт местности //Информатика. – 2019. – №. 2 (02). – С. 86-93.
4. ГИС-ПАНОРАМА – Граф дорог [Электронный ресурс] URL: https://help.gisserver.ru/ru/panagro/index.html?roadgraph.html (дата обращения 10.01.22)
5. Судакова О. В. ИНТЕЛЛЕКТУАЛЬНАЯ ИНТЕРАКТИВНАЯ СИСТЕМА ПЛАНИРОВАНИЯ МАРШРУТОВ СЛУЖБ ДОСТАВКИ, ОСНОВАННАЯ НА ЗНАНИЯХ ПОЛЬЗОВАТЕЛЕЙ. РАЗРАБОТКА ГРАФА ДОРОГ //ИНФОРМАЦИОННЫЕ ТЕХНОЛОГИИ В УПРАВЛЕНИИ, АВТОМАТИЗАЦИИ И МЕХАТРОНИКЕ. – 2021. – С. 259-264.
6. Mangalam K. et al. From goals, waypoints & paths to long term human trajectory forecasting //Proceedings of the IEEE/CVF International Conference on Computer Vision. – 2021. – С. 15233-15242.
7. Toma A. I. et al. Waypoint Planning Networks //arXiv preprint arXiv:2105.00312. – 2021.
8. Pettré J., Grillon H., Thalmann D. Crowds of moving objects: Navigation planning and simulation //ACM SIGGRAPH 2008 classes. – 2008. – С. 1-7.
9. Pettré J. et al. Real-time navigating crowds: scalable simulation and rendering //Computer Animation and Virtual Worlds. – 2006. – Т. 17. – №. 3-4. – С. 445-455.
10. Анатомия KD-Деревьев [Электронный ресурс] URL: https://habr.com/ru/post/312882/ (дата обращения 5.01.22)
11. K-d-дерево [Электронный ресурс] URL: https://ru.wikipedia.org/wiki/K-d-дерево (дата обращения 10.01.22)

12. Choi B. et al. Parallel SAH kD tree construction //High performance graphics. – 2010. – С. 77-86.
13. Местецкий Л. Непрерывная морфология бинарных изображений. Фигуры, скелеты, циркуляры. – Litres, 2018.
14. Erwig M. The graph Voronoi diagram with applications //Networks: An International Journal. – 2000. – Т. 36. – №. 3. – С. 156-163.
15. Ogniewicz R. L., Kübler O. Hierarchic voronoi skeletons //Pattern recognition. – 1995. – Т. 28. – №. 3. – С. 343-359.
16. Мерцалов А. Д. Сравнительный анализ алгоритмов построения скелетов бинарных изображений.
17. Fortune’s algorithm and implementation [Электронный ресурс] URL: http://blog.ivank.net/fortunes-algorithm-and-implementation.html (дата обращения 12.01.22)
18. Fortune S. A sweepline algorithm for Voronoi diagrams //Algorithmica. – 1987. – Т. 2. – №. 1. – С. 153-174.
19. Rachmawati D., Gustin L. Analysis of Dijkstra’s Algorithm and A* Algorithm in Shortest Path Problem //Journal of Physics: Conference Series. – IOP Publishing, 2020. – Т. 1566. – №. 1. – С. 012061.
20. A* Search Algorithm [Электронный ресурс] URL: https://www.geeksforgeeks.org/a-search-algorithm/ (дата обращения 12.01.22)

References

1. GOST 28441-99. Cartography. Terms and definitions / Introduced by the State Standard of Russia. - Instead of GOST 28441-90. - Moscow: IPK Publishing House of Standards, 2000. - p. 2. - 15 p. - (GOST).
2. Tsvetkov V. Digital maps and digital models //The electronic version is posted on the website www. rae. ru. - 2000. - p. 348.
3. Ablameyko S. V., Kryuchkov A. N. Information technologies for creating and updating digital and electronic maps of the area //Computer science. – 2019. – №. 2 (02). – pp. 86-93.
4. GIS-PANORAMA - Graph of roads [E-resource] URL: https://help .gisserver.ru/ru/panagro/index.html?roadgraph.html (Address date 10.01.22)
5. Sudakova O. V. INTELLIGENT INTERACTIVE ROUTE PLANNING SYSTEM FOR DELIVERY SERVICES BASED ON USER KNOWLEDGE. ROAD GRAPH DEVELOPMENT //INFORMATION TECHNOLOGIES IN MANAGEMENT, AUTOMATION AND MECHATRONICS. - 2021. - pp. 259-264.
6. Mangalam K. et al. From goals, waypoints & paths to long term human trajectory forecasting //Proceedings of the IEEE/CVF International Conference on Computer Vision. – 2021. – pp. 15233-15242.
7. Toma A. I. et al. Waypoint Planning Networks //arXiv preprint arXiv:2105.00312. – 2021.
8. Pettré J., Grillon H., Thalmann D. Crowds of moving objects: Navigation planning and simulation //ACM SIGGRAPH 2008 classes. – 2008. – pp. 1-7.
9. Pettré J. et al. Real-time navigating crowds: scalable simulation and rendering //Computer Animation and Virtual Worlds. – 2006. – Т. 17. – №. 3-4. – pp. 445-455.
10. Anatomy of KD-Trees [E-resource] URL: https://habr.com/ru/post/312882 / (Address date 5.01.22)
11. K-d-tree [E-resource] URL: https://ru.wikipedia.org/wiki/K-d-дерево (Address date 10.01.22)
12. Choi B. et al. Parallel SAH kD tree construction //High performance graphics. – 2010. – pp. 77-86.
13. Mestetsky L. Continuous morphology of binary images. Figures, skeletons, circulars. – Litres, 2018.

14. Erwig M. The graph Voronoi diagram with applications //Networks: An International Journal. – 2000. – Т. 36. – №. 3. – pp. 156-163.
15. Ogniewicz R. L., Kübler O. Hierarchic voronoi skeletons //Pattern recognition. – 1995. – Т. 28. – №. 3. – pp. 343-359.
16. Mertsalov A.D. Comparative analysis of algorithms for constructing skeletons of binary images.
17. Fortune's algorithm and implementation [E-resource] URL: http://blog.ivank.net/fortunes-algorithm-and-implementation.html (accessed 12.01.22)
18. Fortune S. A sweepline algorithm for Voronoi diagrams //Algorithmica. – 1987. – Т. 2. – №. 1. – pp. 153-174.
19. Rachmawati D., Gustin L. Analysis of Dijkstra's Algorithm and A* Algorithm in Shortest Path Problem //Journal of Physics: Conference Series. – IOP Publishing, 2020. – Т. 1566. – №. 1. – P. 012061.
20. A* Search Algorithm [E-resource] URL: https://www.geeksforgeeks.org/a-search-algorithm / (Address date 12.01.22)