Лабораторная работа 6

1. Используя алгоритм Дейкстры, реализуйте функцию планирования пути по сеточной 2D карте.

   На вход функции подаются:

   - координаты начальной и конечной клеток
   - карта в виде логического массива, каждый элемент которого обозначает свободную клетку (`false`) или препятствие (`true`).

   В структуре кода используются следующие массивы:

   - `map` — массив, который отражает текущее состояние каждой клетки в зависимости от численного значения. Клетка может находиться в свободном пространстве или быть препятствием, она также может быть уже посещенной или находиться в списке рассматриваемых.
   - `distanceFromStart` — массив, который для каждой клетки хранит наименьшее значение расстояния от начала.
   - `parent` — массив, который для каждой клетки хранит линейный индекс ее родителя, реализующего наименьшее расстояние до этой клетки.

   В проекте `Dijkstra_starter` закончите функцию `DijkstraGrid`. На каждой итерации цикла необходимо найти не посещенные клетки с наименьшим значением расстояния от начала, рассмотреть для каждой из них четырех соседей (с севера, юга, востока и запада) и для этих соседей обновить (если нужно) соответствующие элементы массивов `Map`, `DistanceFromStart` и `Parent`. Также необходимо следить за общим числом клеток, задействованных в поиске (посещенных).

   Протестируйте функцию, запустив файл `TestScript_Dijkstra.m` для различных входных данных.

2. Используя алгоритм A*, реализуйте функцию планирования пути по сеточной 2D карте.

   На вход функции подаются:

   - координаты начальной и конечной клеток
   - карта в виде логического массива, каждый элемент которого обозначает свободную клетку (`false`) или препятствие (`true`).

   В структуре кода используются следующие массивы:

   - `map` — массив, который отражает текущее состояние каждой клетки в зависимости от численного значения. Клетка может находиться в свободном пространстве или быть препятствием, она также может быть уже посещенной или находиться в списке рассматриваемых.
   - `g` — массив, который для каждой клетки хранит текущую оценку расстояния от начала.
   - `H` — массив, который для каждой клетки хранит значение функции Эвристики.
   - `f` — массив, который для каждой клетки хранит сумму значений функции `g` и функции `H`.
   - `parent` — массив, который для каждой клетки хранит линейный индекс ее родителя, реализующего наименьшее расстояние до этой клетки.

   В проекте `AStar_starter` закончите функцию `AStarGrid`. На каждой итерации цикла необходимо найти не посещенные клетки с наименьшим значением `f`, рассмотреть для каждой из них четырех соседей (с севера, юга, востока и запада) и для этих соседей обновить (если нужно) соответствующие элементы массивов `g`, `f`, `Map` и `Parent`. Также необходимо следить за общим числом клеток, задействованных в поиске (посещенных).

   Протестируйте функцию, запустив файл `TestScript_AStar.m` для различных входных данных.

3. Напишите функцию, которая осуществляет проверку, пересекаются ли два треугольника (в плоскости).

   На вход функции подаются два массива 3×2 — `P1` и `P2`, элементы которых (построчно) соответствуют координатам (x, y) вершин треугольников.

   Функция должна возвращать `true`, если треугольники пересекаются, и `false` в противном случае.

   Один из возможных способов — проверить для каждого из 6 образуемых ребер, может ли оно быть «границей», такой что оставшаяся вершина этого же треугольника лежит с одной стороны от ребра, а все вершины другого треугольника лежат с другой стороны от него.

   Протестируйте функцию, вызвав ее для различных входных данных.

4. Используя искусственное поле потенциалов, реализуйте функцию, которая выводит робота из одного положения в другое в двумерном пространстве конфигураций.

   На вход функции подаются:

   - `f` — массив, каждый элемент которого хранит значение функции потенциалов в конкретной точке.
   - `start_coords` и `end_coords` — массивы координат (x, y) начального и конечного положения.
   - `max_its` — максимально возможное число итераций.

   Функция должна возвращать массив `route` из двух столбцов, демонстрирующий изменение координат (x, y) робота по мере прохождения пути до целевой точки. Первая и последняя строки массива должны совпадать с начальным и конечным положением робота соответственно.

   В проекте `PotentialField_starter` закончите функцию `GradientBasedPlanner`. На каждой итерации цикла необходимо обновить положение робота в соответствии со значениями градиента функции `f`, которые хранятся в массивах `gx` и `gy` (не забудьте нормировать градиент). После этого обновите массив `route`, добавив в его конец получившиеся координаты нового положения робота. Расстояние между последовательными локациями не должно превосходить 1.0. Продолжайте описанную процедуру, пока расстояние между текущим и целевым положением робота не станет меньше 2.0, или пока число итераций не достигнет заданного порогового значения `max_its`.

   Протестируйте функцию, запустив файл `PotentialFieldScript.m` для различных входных данных (вы можете изменить, например, положение препятствий или начальное положение робота).