任务书

课程

题目 自动导引搬运车（AGV）路径规划算法的仿真与实现

专业 姓名 学号

主要内容：

随机生成18-20个障碍物，每个障碍物随机占据6-20个栅格，利用常用路径规划方法（以下推荐方法任选其一或自行提出新方法均可），实现AGV从左下到右上的路径规划。

常用路径规划方法推荐：神经网络方法、人工势场法、蜂群算法、遗传算法、蚁群算法、粒子群算法、栅格建模法、深度强化学习算法、基于Petri网的方法、排队理论、图论法、几何法、A\*算法、Dijkstra算法、Floyd算法等。

参考资料：

学生 （签名）

指导教师 （签名）

2020年 10 月 19 日

1. **系统构成**

本实验采用matlab使用栅格法进行常用路径规划算法（dijkstra算法）的实现以及可视化。本仿真系统主要由五个函数组成。第一个函数PlotGrid(map,start,goal)输入栅格矩阵，起始点，终止点得到包含终止点和起始点的二维栅格地图，第二个函数[x y]=Get\_xy(distance,path,map)，由路线的距离，具体路线，地图，得到具体路线的坐标，第三个函数Plot(distance,x,y)由路线的距离，具体路线的坐标在二维栅格图上画出具体路线，第四个函数W=G2D(map)由地图坐标矩阵得到邻接矩阵以供dijkstra算法使用，第五个函数

[Cost, Route] = Dijkstras( Graph, SourceNode, TerminalNode )是dijkstra的matlab实现，由节点地图，源节点和目标节点得到最短路径的值，具体路线，是整个实验的核心。最后为了满足随机生成18-20个障碍物，每个障碍物随机占据6-20个栅格的要求有创建了一个地图矩阵生成函数[Map]=creMap()，来生成一个符合要求的随机地图矩阵。

1. **算法原理**

Dijkstra算法是一个单源最短路径算法，复杂度为O（n2），可以求解非负权重下的最短路径，本次机器人路径规划采用此算法。

问题描述：在无向图 G=(V,E) 中，假设每条边 E[i] 的长度为 w[i]，找到由顶点 V0 到其余各点的最短路径。（单源最短路径）

 2.算法描述

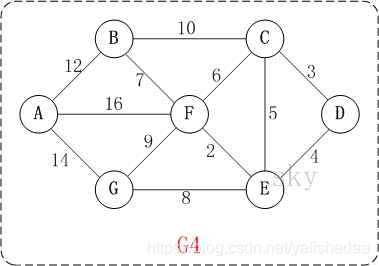
1)算法思想：设G=(V,E)是一个带权有向图，把图中顶点集合V分成两组，第一组为已求出最短路径的顶点集合（用S表示，初始时S中只有一个源点，以后每求得一条最短路径 , 就将加入到集合S中，直到全部顶点都加入到S中，算法就结束了），第二组为其余未确定最短路径的顶点集合（用U表示），按最短路径长度的递增次序依次把第二组的顶点加入S中。在加入的过程中，总保持从源点v到S中各顶点的最短路径长度不大于从源点v到U中任何顶点的最短路径长度。此外，每个顶点对应一个距离，S中的顶点的距离就是从v到此顶点的最短路径长度，U中的顶点的距离，是从v到此顶点只包括S中的顶点为中间顶点的当前最短路径长度。

(1) 初始时，S只包含起点s；U包含除s外的其他顶点，且U中顶点的距离为"起点s到该顶点的距离"[例如，U中顶点v的距离为(s,v)的长度，然后s和v不相邻，则v的距离为∞]。

(2) 从U中选出"距离最短的顶点k"，并将顶点k加入到S中；同时，从U中移除顶点k。

(3) 更新U中各个顶点到起点s的距离。之所以更新U中顶点的距离，是由于上一步中确定了k是求出最短路径的顶点，从而可以利用k来更新其它顶点的距离；例如，(s,v)的距离可能大于(s,k)+(k,v)的距离。

(4) 重复步骤(2)和(3)，直到遍历完所有顶点。



以上图G4为例，来对迪杰斯特拉进行算法演示(以第4个顶点D为起点)。

**初始状态**：S是已计算出最短路径的顶点集合，U是未计算除最短路径的顶点的集合！  
**第1步**：将顶点D加入到S中。  
    此时，S={D(0)}, U={A(∞),B(∞),C(3),E(4),F(∞),G(∞)}。     注:C(3)表示C到起点D的距离是3。

**第2步**：将顶点C加入到S中。  
    上一步操作之后，U中顶点C到起点D的距离最短；因此，将C加入到S中，同时更新U中顶点的距离。以顶点F为例，之前F到D的距离为∞；但是将C加入到S之后，F到D的距离为9=(F,C)+(C,D)。  
    此时，S={D(0),C(3)}, U={A(∞),B(13),E(4),F(9),G(∞)}。

**第3步**：将顶点E加入到S中。  
    上一步操作之后，U中顶点E到起点D的距离最短；因此，将E加入到S中，同时更新U中顶点的距离。还是以顶点F为例，之前F到D的距离为9；但是将E加入到S之后，F到D的距离为6=(F,E)+(E,D)。  
    此时，S={D(0),C(3),E(4)}, U={A(∞),B(13),F(6),G(12)}。

**第4步**：将顶点F加入到S中。  
    此时，S={D(0),C(3),E(4),F(6)}, U={A(22),B(13),G(12)}。

**第5步**：将顶点G加入到S中。  
    此时，S={D(0),C(3),E(4),F(6),G(12)}, U={A(22),B(13)}。

**第6步**：将顶点B加入到S中。  
    此时，S={D(0),C(3),E(4),F(6),G(12),B(13)}, U={A(22)}。

**第7步**：将顶点A加入到S中。  
    此时，S={D(0),C(3),E(4),F(6),G(12),B(13),A(22)}。

此时，起点D到各个顶点的最短距离就计算出来了：**A(22) B(13) C(3) D(0) E(4) F(6) G(12)**。

1. **实验结果**

Debug函数如下：

clear;

clc;

close all;

[map]=creMap()%Map采用的是按要求生成18-20个障碍物，每个障碍物占6-16格

start=[1 100]%起点的坐标

goal=[100 1]%终点的坐标

Dmap=G2D(map);%生成栅格地图的邻接矩阵

PlotGrid(map,start,goal)%画出二维栅格地图

[distance,route]=Dijkstras(Dmap,9901,100)%由算法得到最短距离和具体路先

[x y]=Get\_xy(distance,route,map);%将具体路线用栅格坐标表示出来

Plot(distance,x,y)%在栅格图上画出具体路线

Map采用的是随机逻辑数组（由0，1构成）其中0的个数大于40%。采用此种方法有大概率由于最短路径不存在（即0的个数出现在起始点或终止点或由于障碍物阻碍导致不能到达）导致无法生成结果，再次运行即可。

随机运行结果展示：

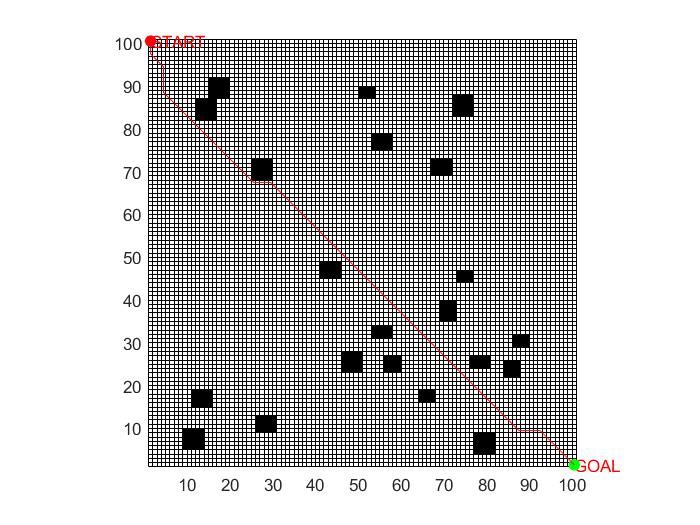
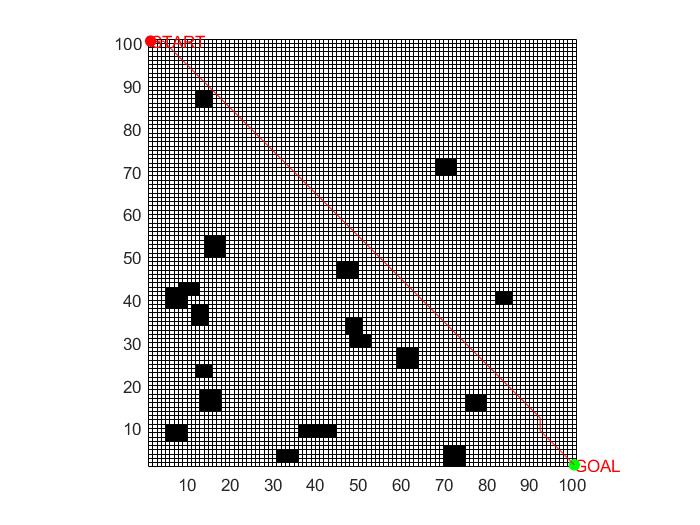


图1随机运行结果1



图二 随机运行结果2

**四、数据结果分析及后续展望**

Dijkstra算法可以有效的算出点到点的最短路径，并反馈给机器人进行路径规划，此中方法用于实践需要先得到环境地图，才能进行规划，且在拐点数量与其他算法如蚁群算法，人工势场法及其改进算法相比并不占优势，拐点数量的增加必然会导致机器人移动的连续性和速度。Dijkstra 算法核心是以某个顶点为起点逐渐向外扩展，直到达到目标顶点为止，每次扩展都会尝试路径中所有相邻的顶点。虽然该算法能得到最短路径的最优解，但是它要计算所有点之间的路径，进而运算的效率比较低。由前面的dijkstra算法原理可知，其时间复杂度为O（n2）、A\*算法的时间复杂度为O（n）、floyd算法为O（n3）。但与其相比A\*算法仍未脱离贪心算法的范畴，并且往往也会同样陷入到产生局部最优的陷阱与问题中，在确定启发函数 h(n)时也存在着相应的难度与问题。