交叉项目综合训练大作业

李明 自 85 2018011546

摘要

移动机器人的动态避障要求机器人在机器人运动过程中实时根据当前检测到的障碍信息和当前的位置信息给出到目标点的路径轨迹。在动态避障方面，难点在于变化的环境信息无法通过事先给定的方式给出，这给机器人的路径规划带来了很大的困扰。本文将介绍相关的路径规划算法以及动态避障方法，其中着重给出人工势场法的实现原理及过程，并给出人工势场法路径规划的过程。

摘要： 移动机器人；路径规划；动态避障；人工势场法

引言

路径规划问题比较常用的方法有 Dijkstra 算法、A^*算法、Hybrid A^*算法，这些方法的特点是需要知道全局的地图信息，在有了全局地图信息的基础上进行搜索，从而找到一条到达终点的路径，很难用于动态避障；在上面的算法的基础上，为了适应动态环境实时计算路径所需要的计算量问题，有基于A^*算法的LPA^*的增量启发式搜索算法；但是上面的方法中都需要已知全局地图信息，这在许多时候并不可行，人工势场法可以用以解决当环境信息不是已知时的动态避障，本文将对这种方法进行介绍，也会在后面给出具体实现的例子和实现结果。

路径规划算法

1 Dijkstra 算法

Dijkstra 算法是一种经典的图搜索算法，使用 Dijkstra 算法可以求解一个点到其余各个顶点的最短路径。这种算法是以起始点为中心，将起点加入到初始集合 S 中。接下来，对于每一次迭代，找到距离初始集合中最近的点，更新其距离，并将其加入到初始集合 S 中，直到图中所有的点都在 S 中，则找到了每一个点到 S 的最短路径长度及最短路径。在路径规划中，并不需要找到所有的点到初始点的距离，只需要 S 包含终点时，便可以停止迭代。

Dijkstra 算法的不足是需要提前知道全局地图信息，才能以图的方式进行搜索，这也限制了其在动态避障中的应用。

2 A^*算法

A^*算法是一种启发式搜索算法，这种方法通过定义了启发函数$f(m) = g(m) + h(m)$其中$g(m)$表示从初始点到 M 点的代价，而$h(m)$是从 M 到目标点的估计距离。A^*算法既利用了已经所过的图的信息，又加入了对未来状况的预测，既能够保证最优性，又能够比较快速的搜索到目标。A^*算法在实现时，需要维护 open 表和 close 表，通过计算 open 表中的代价选择下一步要搜索的节点，直到目标点加入到 open 表中。

A^*算法同样是一种需要直到全局地图信息的搜索方法，而且搜索需要的存储空间比较大，规划的路径不平滑，不适合车辆、机器人等直接驾驶。

3 $Hybrid\ A^*$算法

$Hybrid\ A^*$方法是在A^*算法的基础上考虑物体实际运动约束的方法。在$Hybrid\ A^*$算法中，不禁需要考虑物体的位置，还需要考虑物体的方向包括转向角的改变和是否倒车等。并且与A^*算法不同的是，在$Hybrid\ A^*$中，物体不一定出现在规划的格点中心。

$Hybrid\ A^*$算法同样是一种需要直到全局地图信息的搜索方法。

4 LPA^*算法

LPA^*算法一种基于A^*算法的增量启发式搜索算法，以起始点为搜索起点，按照 Key 值大小进行搜索，Key 值包含启发式函数 h，作为启发原则来影响搜索方向。除了应用在静态地图中，LPA^*可以用于动态地图中。这并非是不需要全局信息，而是LPA^*可以利用之前已经计算的地图距离信息，快速地重新进行规划路径。

5 人工势场法

这一部分，我们将着重介绍人工势场法。

a. 人工势场法问题的假设

移动机器人假设：我们假设机器人可以具有一定的视野范围，可以观测到距离自己半径为 r 内是否有障碍。

地图假设：我们在这里不要求地图是静态地图，地图中的障碍可以随着时间改变。

机器人位置假设：机器人能够获取到自己在地图中的位置信息以及目标点的位置

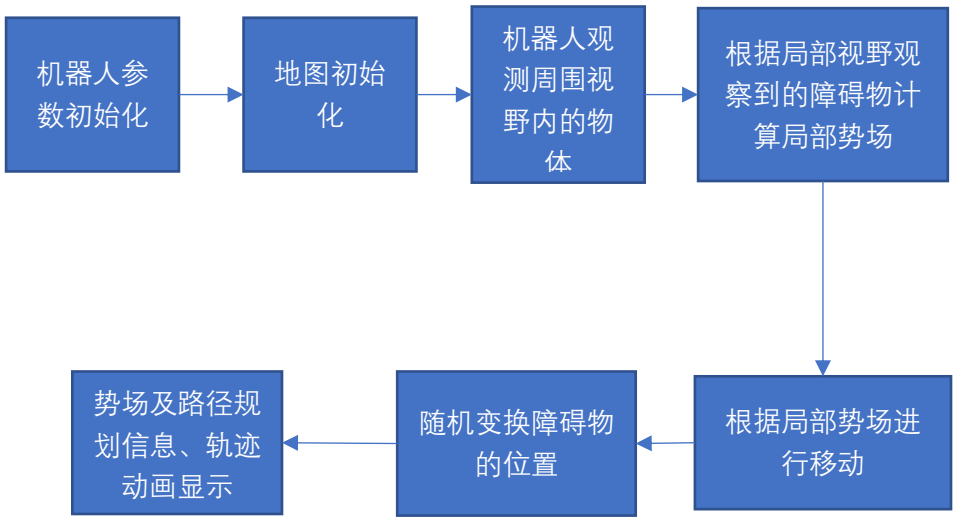
b. 人工势场的基本原理

在人工势场中，目标点具有吸引力，会吸引机器人前往目标点，而障碍具有排斥力，会使得机器人远离障碍。吸引力和排斥力的大小与距离有关，在本问题中假设与距离的二次方成反比，所以势能与距离成反比。则可以知道，当不存在障碍时，机器人将因为吸引力以直线前往目标点。

c. 人工势场法的实现细节

在地图的表示中，我们将地图进行栅格化，在栅格化后的地图上表示出机器人、起始点、目标点以及障碍物。

整个实现的框架如下所示



下面分别介绍地图与机器人的相关功能。

1) 机器人参数初始化

需要确定机器人参数的初始位置、能够观测的到的视野半径（在本问题中，采用了以机器人为中心、以二倍半径为边长的举行视野）以及目标点的位置。

2) 地图初始化

地图初始化中包含地图的大小、起始点、目标点的设置以及障碍的设置。

3) 机器人观测周围环境

在机器人初始化时已经给出了机器人的视野范围，将机器人放入地图之中，地图会给机器人反馈，以机器人为中心，以机器人观测半径为限，给出机器人所在区域的局部地图。

4) 根据局部地图计算势场

这里的局部势场是以机器人观察到的障碍和目标点构成的，这样的方法是与实际环境相符合的，即机器人在动态环境也就是环境地图未知的情况下进行路径规划和避障，是只能根据观测到的障碍信息进行相关的判断。

5) 根据局部势场进行移动

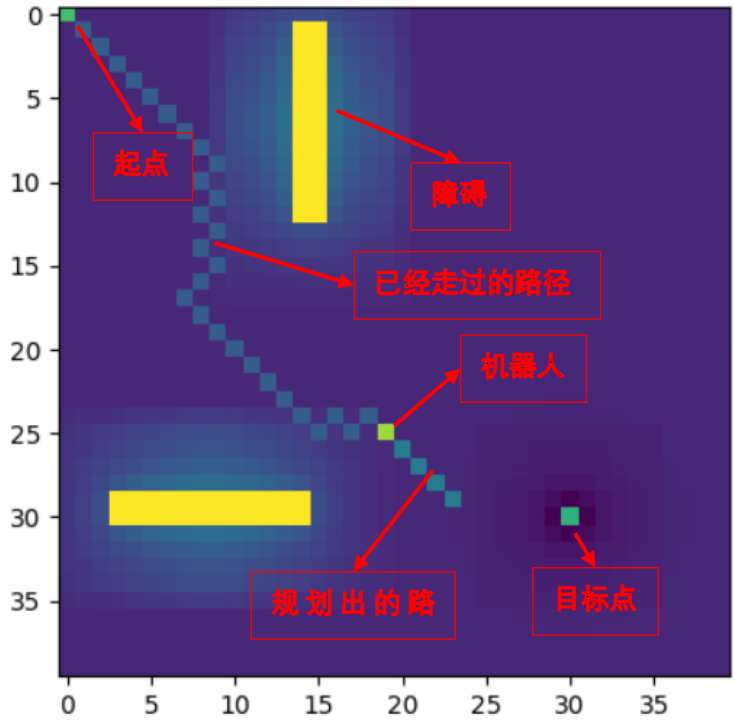
在做移动时，机器人会根据局部势场中每个格点的势能大小，按照八近邻（即相连的周围的八个格子）寻找势能更低的点，将其加入到下一步要走的路中，完成局部路径规划。在每次运动时行走一步。需要注意，由于机器人的视野范围有限，所以当机器人在相邻的两步之间的势场变化会因为突然看到或者突然看不到障碍物而比较大，这也会导致机器人在做局部路径规划时，相邻两次的方向可能不同，但是组合起来又是朝着正确的方向行走。

6) 随机变换障碍物的位置

为了模拟障碍物的动态变化，我对障碍物增加了随机变换位置的功能，障碍物会在原来的基础上以 1/2 的概率进行变换位置，机器人需要在动态的地图中找到合理的路径。

7) 显示信息

在这个问题中，我用不同的颜色表示在整个地图中的实际的势场信息（注意，这里的势场是所有的障碍的势场，而不是局部地图的势场）、机器人的位置、路径规划信息、起点位置、目标点位置、障碍位置等元素，如下图所示



d. 实现技巧与难点

1) 封装

在本问题中，涉及的对象很多，如果在一个主函数中将各个变量定义出来，会将代码变得很长、并且难以找到 bug。因此，我采用了类和函数的封装来解决这个问题。我封装了机器人 MyRobot 类、地图 MyMap 类、计算两点之间的能量 EnergyControl 类。

```
# 地图标志类
class MapMarker:···

# 能量计算类
class EnergyControl:···

# 机器人类
class MyRobot:···

# 地图类
class MyMap:···

# 获取机器人能观测到的局部地图
def getRobotRoundMap(map:MyMap,robot:MyRobot):···

# 生成障碍物
def generateObstacle(info=0):···
# 生成起点
def generateStart():···
# 生成终点
def genetateTarget():···
```

封装的函数和类

2) 可视化

为了将结果进行展示，我将各个关键变量包括全局势场信息、机器人的位置、路径规划信息、起点位置、目标点位置、障碍位置等等，通过调整不同元素的颜色参数，达到可视化的目的。

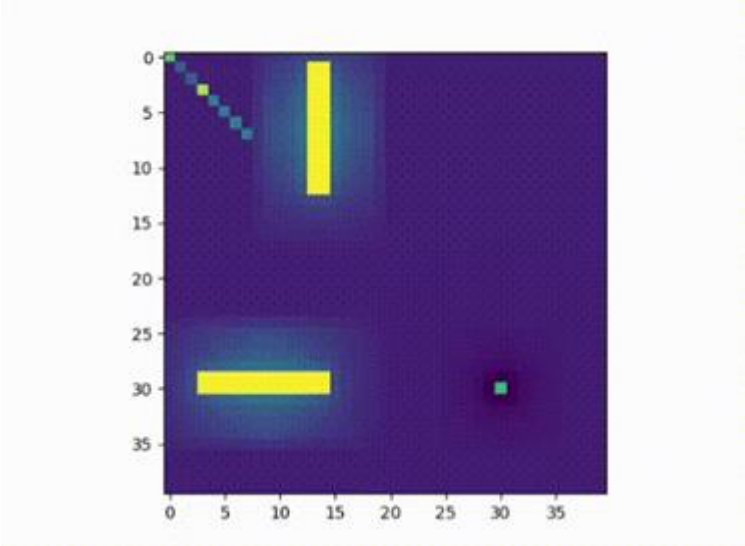
可视化占了比较多的运算时间，程序中的主要时间都是在执行可视化的显示操作中的全局势场的计算中。

```
class MapMarker:
    OBSTACLE = 300
    TAGERT = 180
    START = 200
    SPACE = 0
    ROBOT = 250
    ROBOT_TRACE = 60
    PLAN_TRACE = 100
```

定义的不同类型的地图标志定义

e. 实现结果

使用人工势场法进行动态障碍物的探测与躲避，得到如下的动态避障演示图（点击视频打开）

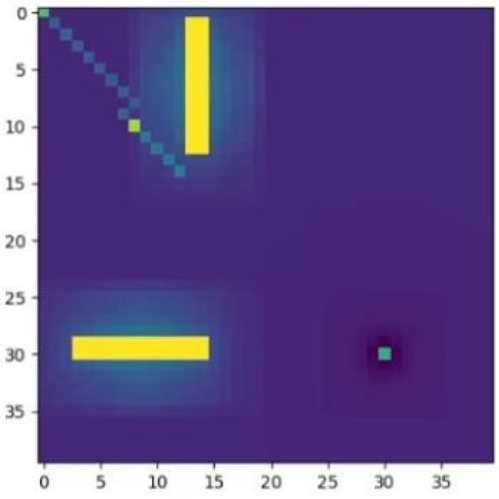


f. 算法的优点与不足分析

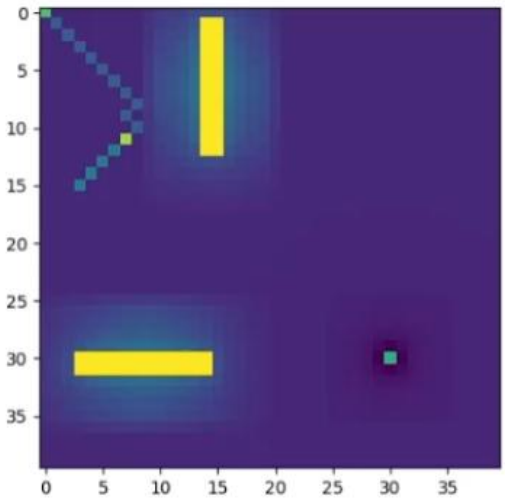
人工势场方法考虑了机器人的视野范围的限制，能够根据视野内观测到的障碍物进行计算局部势场和路径规划，能够很好的达到动态避障的目标。

人工势场方法需要设置较为合理的障碍和目标点的势能，这样才能比较好的达到效果。另外，在建模本身，由于机器人固定了视野范围，可能会出现在运动时观测到的势场快速改变的情况，比如下面两幅图中的情况，第一幅图是由于上一步由于视野半径小于到

障碍的距离而做出了向障碍方向移动的决策，同时规划出的路径也是没有考虑到障碍；而第二幅图中，由于当前位置可以观测到障碍，由于障碍的斥力作用，路径规划出了向左的运动，但是其实，机器人本身也向左移动；这样的情况在接下来的回合中也可以看到。虽然不影响最终能够到达目标点，但是者说明建模本身并不够完美，才给出了这样具有很大变化性的结果。当然，如果我们将全局的地图信息都交给机器人，可以规划出更好的结果，但是这样其实并不符合动态障碍的要求，动态障碍本身决定了地图信息只能够通过局部观测来得到，从这一方面来看，我们给出的方法也是更符合实际的方法，只不过在观测范围上还可以优化，比如当在一个方向上的视野范围是能够观测到的最近的障碍，但是这样也会大大增加计算量，对仿真模型的性能实现提出了更高的要求。



上一步未发现障碍而移动的路径规划结果



上一步发现障碍而移动的路径规划结果

总结

本文介绍了使用比较多的路径规划和动态避障的相关算法，对这些算法的基本原理和实现过程做了简要介绍。后面本文着重介绍了人工势场法动态避障的方法和原理，并给出了自己的实现。在实现过程中，我利用了类的封装使得代码结构更加易懂和易于维护，并且为了对结果进行更加明显地显示出来，做了界面的可视化处理，使得结果能够以动画视频的方式输出，对用户更加友好。