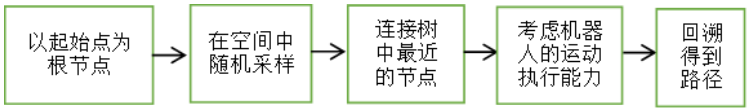
智能移动技术报告

1 实验方法

1.1 RRT 路径规划

1.1.1 原理描述

RRT 是一种基于概率的路径规划方法，它的基本思想是采用树形结构存储路径，通过随机采样并连接最近节点的方式拓展节点。它的优点是能够迅速地找到一条可行路径，便于后续的轨迹规划。



1.1.2 实验中遇到的问题和解决办法

问题：在实验中，一般 RRT 都能较快地收敛得到结果；但有时候算法可能经过较长时间还未能收敛。

原因分析：在测试中，我们发现，这与每次拓展新节点的步长和采样点的方向有有关。如果每次拓展时的步长较短，则需要许多次迭代才能收敛，同时节点数量会很多；如果完全依赖随机采样，会存在较多树枝是背离目标点方向增长的，也会需要很长时间才能收敛。

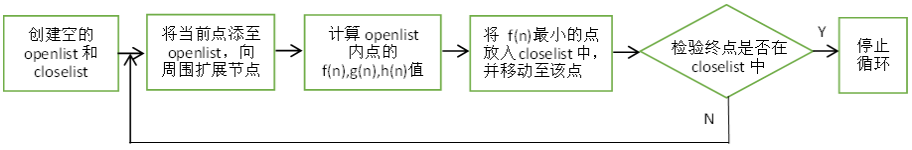
问题解决：针对第一个原因，我们采用了变步长的方式拓展节点，当当前节点距离目标点较远的时候，适当增大每次拓展节点的步长，随着距离减小，每次拓展的步长也随之减小；针对第二个原因，我们设定了每次随机采样有 20%的概率会选取目标点，这样就会使得搜索树树更快地向目标点拓展。

1.2 路径规划法：A*

1.2.1 原理描述

在该例中，我们将小车上下左右、斜方向的八个点作为扩展节点，并通过改变小车单步步长的方法来优化路径搜索的效果。关于判断是否将扩展节点加入 openlist，我们的标准是：若该节点已经在 openlist 中则忽略，若该点会与障碍物发生碰撞则忽略，将其余点都加入 openlist 中。为了让小车更快地完成路径规划，我们将“终点在 closelist 中”的条件改为“小车与终点的距离 < 小车步长+10”。

经调试，小车步长为 60 时路径规划效果较好，能在 1s 内完成。



1.2.2 实验中遇到的问题和解决办法

1.2.2.1 关于小车步长

小车步长 ＜40 时，一旦搜索路径经过两个间距较小的障碍物，就会在障碍物间来回搜索而无法迅速结束该条路线的搜索；步长>70 时，小车容易误判与障碍物发生了碰撞，导致原本可行的路线被它摒弃。

1.2.2.2 小车在终点附近往复运动

终止条件为“终点在 closelist 中”时，由于小车步长不为一，小车会在终点附近往复震荡运动。故应当把终止的范围扩展为以终止点为中心、以大于小车步长的值为半径的圆内。

1.3 反馈控制法

1.3.1 原理描述

反馈控制法的基本思想是根据当前状态与目标状态的差异，来生成一个使得差异减小的对（v,w）的控制律。根据当前位置与目标位置的距离ρ，生成速度控制律$v = k_p\rho$；根据当前机器人方向与目标位置连线的夹角α和当前位置与目标位置连线的方向β，生成角速度控制率$\omega = k_\alpha\alpha + \mathrm{k}_\beta\beta$。它的想法比较简单，在设置好参数后也有不错的效果。

1.3.2 实验中遇到的问题和解决办法

问题： 一开始设置的控制率轨迹很容易跑飞，或者要绕很久才能到达路径点。

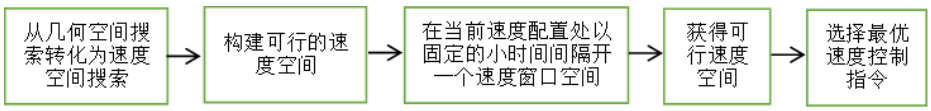
原因分析：跑飞可能是因为控制率的参数设定不当和每次发送命令后间隔时间设定不当。当速度参数过大时可能出现来不及减速就冲过目标点，当角速度参数设定不当可能会出现一直转圈，难以向目标点前进的情况；因为机器人存在加速度和角加速度的限制，每次发送命令后需要一个加减速的时间，这就使得发送的命令与实际执行情况存在一定偏差；

问题解决：最终我们采用 ppt 上的参数$k_p = 3$，$k_\alpha = 8$，$\mathrm{k}_\beta = 1.5$，同时设置最大速度限制为 1.5m/s，这样保证了机器人能有较高速度的同时不会漂走，同时发送命令后设置一个等待时间，使机器人能够更接近命令情况来执行。

1.4 DWA 避障规划

1.4.1 原理描述

动态窗口法(DWA)是一种避障规划方法，它的基本思想是建立一个可行速度空间，计算不同（v, w）情况下的评价函数（朝向目标点、远离障碍物、速度最大化），选择当前时刻下最优的（v, w）。



1.4.2 实验中遇到的问题和解决办法

1.4.2.1 忽略了程序运行需要的时间，并且没有考虑到位姿估计的累计误差

问题：由于最开始我们取的机器人控制时间间隔较短，大概是 0.1s，在发送（v, w）指令后，执行 time.sleep(0.1)，但是运行起来后发现机器人一直在乱跑，调节参数完全不起作用；

原因分析：这主要是由两个部分的原因造成的：首先程序运行需要时间，每两次发送（v, w）指令之间的时间大概有 0.05s 左右(当时)，所以说相当于这部分时间机器人仍然在以之前的速度跑；其次，我们只在最开始读取了机器人的位姿，中间一直是通过计算得到最新的位姿，这就不可避免的存在累计误差，再加上我们忽略了程序运行时机器人还在跑，所以导致机器人的实际位姿和我们计算的的严重不符；

问题解决：于是我们首先增大机器人的控制时间间隔，并加入判断$if(\Delta time < time_interval)\ time.sleep(time_interval - \Delta time)$，以尽量消除程序运行时间的影响；同时我们在每一次发送（v, w）后，重新读取机器人位姿，以进行下一轮（v, w）的选择，以尽量消除累计误差的影响。

1.4.2.2 DWA 可以调节的参数很多，刚开始时调参的效率很低

问题：刚开始对 DWA 进行调参时，无论如何调参，机器人都是刚出发就乱转；

原因分析：DWA 的可调参数很多，且有些参数之间存在联动关系，我们一开始就是忽略了这些关系，只凭机器人跑的结果盲目进行调参，导致效果很不好；

问题解决：通过分析，我们首先发现机器人的速度总是会非常快速的增长到最大，很容易失控，所以先降低了机器人的最大速度（降低最大速度没有和实际情况相冲突），并据此确定了控制时间和预测时间；第三，我们在调参时对评价函数的各个部分进行了打印，以进行观察，结果发现归一化$math.hypot(curPoint - destinationPoints)/math.hypot(maxX - minX, maxY - minY)$后，在不同（v,w）情况下，评价函数的这个部分相差非常小，难以选出最佳（v, w），所以我们将这一部分的归一化去除，并相应的提高评价函数其他部分的权重，经过以上调整后，后续调参逐渐进入正轨。

1.5 RRT / A*路径规划 + DWA 避障规划

1.5.1 基本介绍

RRT / A*+DWA 的规划方法主要针对单纯使用 DWA 偶尔会出现 stuck 的情况进行尝试，如果我们事先规划好路径，以尽量避开那些障碍物密集分布的地方，那么机器人就基本上不容易 stuck。所以我们首先使用 RRT / A*的路径规划方法得到一条安全的路径，然后在其中选择 5~6 个点，再使用 DWA 在这些点之间依次避障规划，最终达到 goal。

2 总结分析

2.1 方法比较

所用方法	static_simple	static_difficult	dynamic_simple	dynamic_difficult
dwa	26.812s	27.532s	26.013s	31.610s
rrt+dwa	39.420s	41.422s	43.024s	51.130s
A*+dwa	27.570s	27.654s	31.225s	29.234s
rrt+反馈控制	28.423s	39.542s	44.690s	42.994s

注：表中的时间为跑 3 轮的平均（每轮往返 5 次），其中为方便记录时间，在 rrt+dwa、A*+dwa，rrt+反馈控制方法中忽略了第一次从起点到目标点的路径规划时间（即以机器人开始移动时作为计时起点，以机器人停止移动时为计时终点）。

2.2 实验结果分析与讨论

（1）DWA 的性能还是比较好的，特别是它在静态障碍物和动态障碍物两种情况下的规划时间是相近的，对动态情况的处理比较好；在一些特殊情况下，它可能会 stuck，我们针对这一情况进行了程序上的调整，但是在极特殊情况下，它仍然会 stuck。

（2）rrt+dwa 和 A*+dwa 的尝试还是比较成功的，在先进行路径规划，再进行避障规划的情况下，stuck 的现象得到明显的改善，再加上路径规划本身需要的时间较短（大概 1~2s），所以在时间上，对于整体的规划效果没有很大的影响。其中 rrt 方法生成的路径经常会绕一大圈，在这一点上 A*算法性能更好。

（3）rrt+反馈控制在静态障碍物的情况下表现还是比较好的，但是受超调的影响，它不可避免的会与原规划路径有一定的偏差，有时候会撞到障碍物；在动态障碍物情况下，我们曾设想在控制率中增加一项与障碍物的距离有关的罚函数，在接近碰撞时减速且转向，但时间关

系未能完全实现。

3 分工贡献

成员姓名	分工	贡献
student1	dwa、dwa+rrt	1/3
student2	A* 、dwa+A*	1/3
student3	rrt、反馈控制法	1/3