** **

**智能系统技术实践实习报告——**

**基于 Matlab/Simulink 的 ROS 移动机器人运动控制**

|  |  |  |
| --- | --- | --- |
| **指导老师** | ： | XXX |
| **姓名** | ： | XXX |
| **班级** | ： | XXXXX |
| **学号** | ： | XXX |

二〇二三年一月

**目录**

[第一章 实验目的和实验内容 1](#_Toc123761411)

[1.1 实验目的 1](#_Toc123761412)

[1.2 实验内容 1](#_Toc123761413)

[第二章 实验原理 1](#_Toc123761414)

[2.1 主要步骤 1](#_Toc123761415)

[2.2 ROS的基本原理 2](#_Toc123761416)

[2.3 路径规划算法 2](#_Toc123761417)

[2.3.1 概率路线图法（PRM） 2](#_Toc123761418)

[2.3.2 快速扩展随机树（RRT） 3](#_Toc123761419)

[2.4 跟踪控制算法 4](#_Toc123761420)

[2.4.1 纯跟踪路径跟踪算法 4](#_Toc123761421)

[2.4.2 Stanley路径跟踪算法 4](#_Toc123761422)

[第三章 实验结果分析 5](#_Toc123761423)

[第四章 总结 7](#_Toc123761424)

[附录 8](#_Toc123761425)

# 实验目的和实验内容

## 实验目的

设计移动机器人路径规划和轨迹跟踪控制算法，使用 MATLAB 或Simulink 进行代码化或图形化编程，在 Gazebo 环境下控制移动机器人，并实现其运动控制目标。

## 实验内容

(1) ROS 的基本原理；

(2) ROS 的通讯机制；

(3) Matlab/Simulink 对 ROS 编程的语法结构；

(4) 移动机器人路径规划算法设计；

(5) 移动机器人轨迹跟踪控制算法设计；

(6) Gazebo 环境下的移动机器人运动控制仿真实验。

# 实验原理

## 主要步骤

(1) 使用 rosinit 命令创建 ROS 环境；

(2) 使用 inflate 命令对地图进行膨胀操作；

(3) 使用 PRM 算法为移动机器人规划可行运动路径；

(4) 为移动机器人创建 PurePursuit 控制器，用于轨迹跟踪控制；

(5) 创建 rospublisher 和 rossubscriber，分别用于向移动机器人发送速度控制指令和从机器人端接收位姿信息；

(6) 使用 rosshutdown 命令结束 ROS 交互。

## ROS的基本原理

ROS可以形象的描述为一个工厂的运行机制，创建好一个工作空间(workspace)就像一个工厂，工厂里又有好多个生产车间，每个功能包（pkg）看作是一个生产车间，每个生产车间又有好多工人在配合，每个节点（node）看作是一个工人，节点是又是可执行程序的最小单位，工人们之间相互沟通通过消息（message）来完成。

话题是单向的，一般用于连续发送数据的传感器，建立一次联系后，一个发布者可以向多个订阅者发送信息，同样，一个订阅者也可以订阅多个发布者的消息。服务是同步双向的通信机制，服务器只有在有请求的时候才响应，客户端在发出请求后才接受响应。当服务的请求和响应完成时，两个连接点自动断开。动作的通讯方式与服务有类似的情况，不同的是服务器收到请求后直至完成响应所需时间较长，中途需要反馈给客户端目前完成的情况，报告当前的现状。

## 路径规划算法

路径规划就是在自由位形空间中为机器人寻找一条能从初始位置运动到目标位置的路径，能够绕开障碍物，安全，快速地到达目标点。路径规划问题分为全局路径规划（又称基于地图的规划，整个周围的环境已知）和局部路径规划（又称动态重规划，它重新计算路径，以避开未知的动态障碍）两类，本实习中涉及的路径规划为全局规划问题。

在实习中我们采用PRM和RRT两种方法规划路径，它们均属于基于采样的路径规划算法，这种算法中，搜索树或路线图是通过在状态空间中随机添加节点来创建的，通常使用启发式方法来探索搜索空间并偏转搜索方向。

### 概率路线图法（PRM）

PRM（Probabilistic Roadmap Method概率路线图法）是一种基于图搜索的方法，它将连续空间转换成离散空间，再利用A\*、Dijstra等搜索算法在路线图上寻找路径，以提高搜索效率。这种方法能用相对少的随机采样点来找到一个解，对多数问题而言，相对少的样本足以覆盖大部分可行的空间，并且找到路径的概率为1（随着采样数增加，P（找到一条路径）指数的趋向于1）。显然，当采样点太少，或者分布不合理时，PRM算法是不完备的，但是随着采用点的增加，也可以达到完备。所以PRM是概率完备且不最优的。

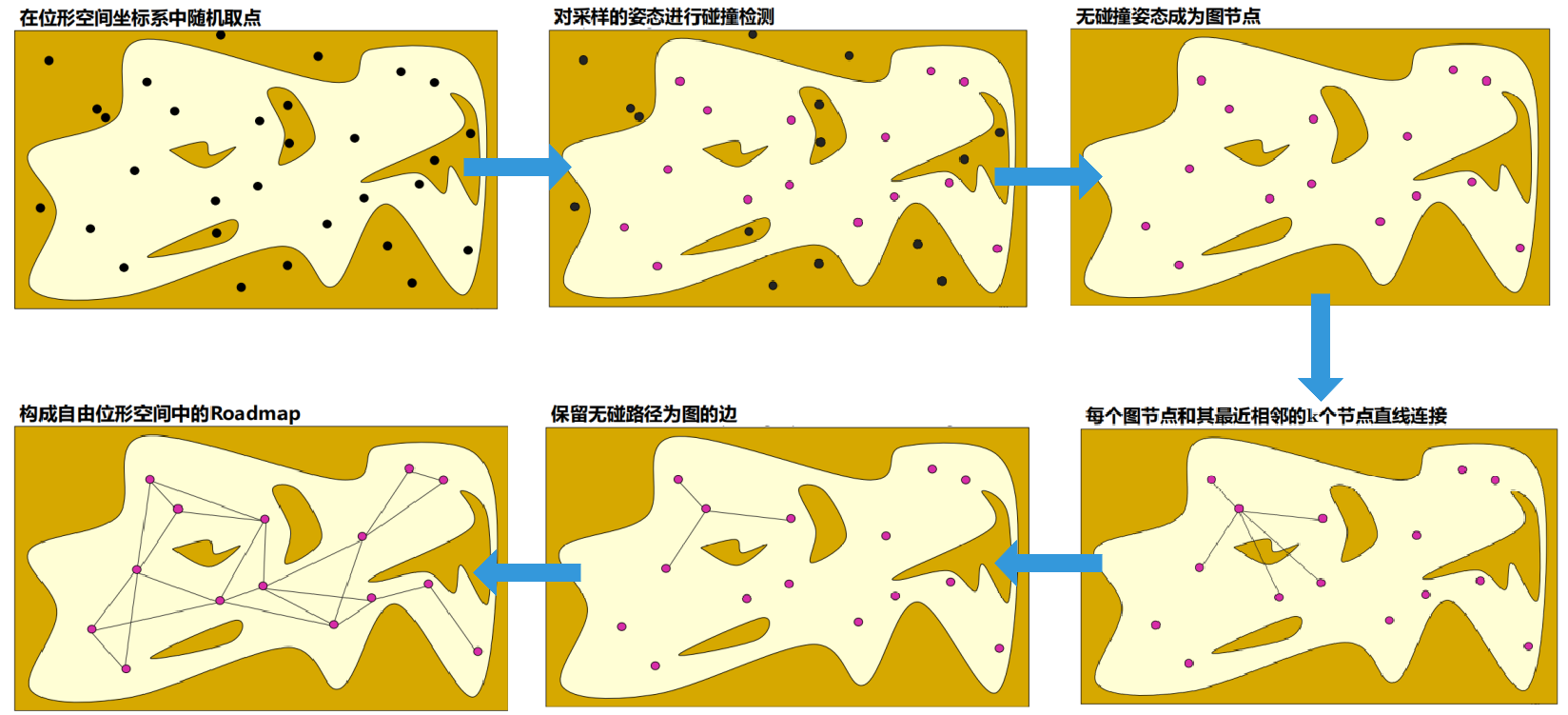


图 2- 1 PRM方法的步骤

可以将PRM算法的步骤分为两大步：构建拓扑连通图、在连通图中搜索最优路径。用于搜索最优路径的算法可分为：

①精确算法：包括深度优先法、广度优先法、优先级定义的广度优先法、Dijstra 算法。存在问题：耗时，难以满足机器人在线快速规划要求

②近似算法：包括各种启发式算法和准启发式算法。启发式搜索算法有 A\*、D\*、 Focused D\*等；准启发式搜索算法有粒子群算法、遗传算法、蚁群优化等。

### 快速扩展随机树（RRT）

RRT（Rapidly-exploring Random Tree快速扩展随机树）是一种常用的基于采样的路径规划方法，用于在连续状态空间中寻找路径。它以一个初始点作为根节点，通过随机采样增加叶子节点的方式，生成一个随机扩展树，当随机树中的叶子节点包含了目标点或进入了目标区域，便可以在随机树中找到一条由从初始点到目标点的路径

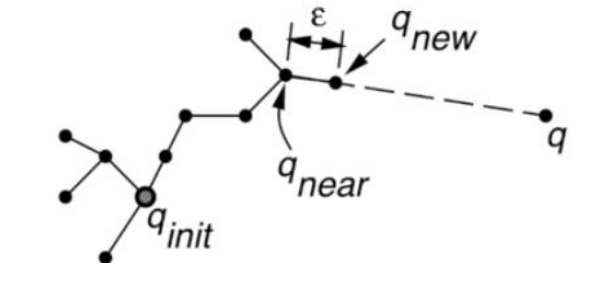


图 2- 2 RRT算法示意图

## 跟踪控制算法

### 纯跟踪路径跟踪算法

Pure Pursuit是一种用于路径跟踪的控制算法。它通过计算角速度控制机器人从当前位置移到机器人前方的某个预瞄点。假定线速度是恒定的，当然可以随意更改机器人的线速度。该算法会根据机器人的当前位置在路径上移动预瞄点，直到路径的终点。可以想象成机器人不断追逐它前面的一个点。参数LookAheadDistance就决定将预瞄点放置有多远。

Pure Pursuit并不是传统的控制器，而是充当着跟踪算法的角色，用于跟随路径，航向点的信息是唯一的，可以指定所需的线速度及最大角速度。给定机器人的位姿（位置和方向）作为输入，Pure Pursuit会计算机器人的线速度和角速度命令。机器人如何使用这些命令取决于你所使用的系统，因此请考虑给定这些命令的机器人如何执行动作。LookAheadDistance参数它告诉机器人在跟踪路径上要走多远

### Stanley路径跟踪算法

Stanley控制算法是一种基于横向误差（cross-track error）的跟踪算法。在Stanley算法中，我们同样需要使用到阿克曼转向模型，其中（Px，Py）是位于跟踪路径上的距离前轮中心最近的路径点，d是表示前轮中心到路径点的距离，也就是距离偏差。在不考虑角度偏差的情况下，距离偏差越大，前轮转向角越大。

# 实验结果分析

王老师所给的例程中使用的是PRM算法进行路径规划，纯跟踪算法进行路径跟踪。在学习并理解了PRM方法原理的基础上，我尝试采用了RRT进行路径规划，并成功完成了实验。两种方法的结果如下：

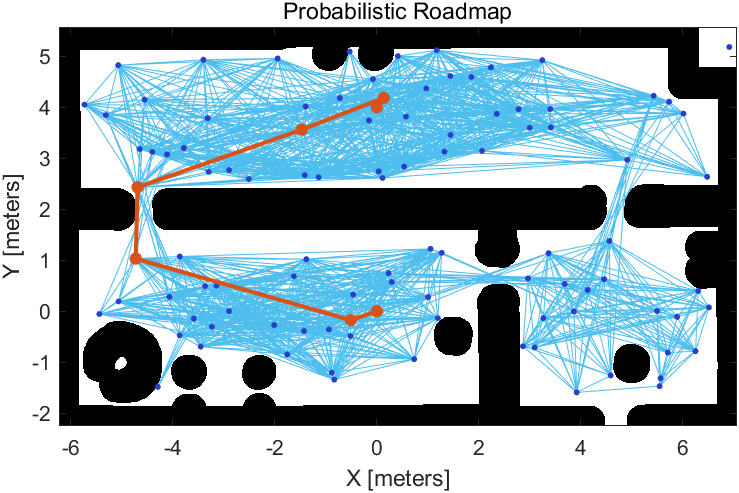


图 3- 1 PRM路径规划结果

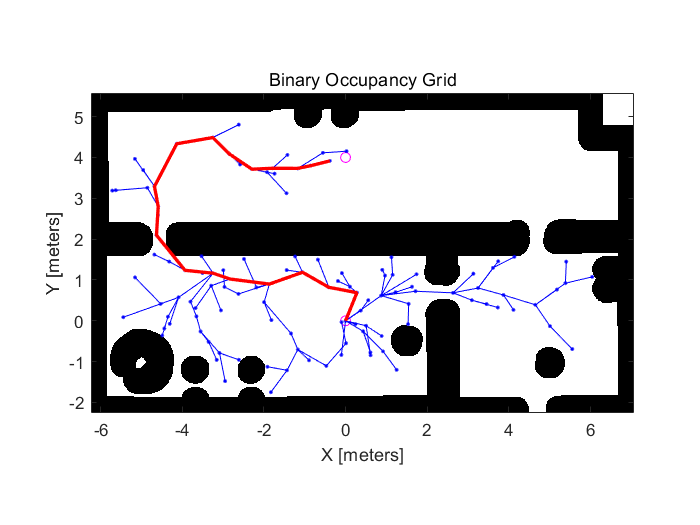


图 3- 2 PRM路径规划结果

从上面两张图中可以直观地看到PRM和RRT两种方法在构建拓扑连通图时的区别，一个是先在地图上均匀地随机撒点，去除与障碍物重合的点，然后把所有互相可视的点两两连接起来，组成的连通图；而另一个则是把初始点作为根节点，然后随机采样增加叶子节点，从而生成一个随机扩展树，这棵树就是连通图，后续搜索最优路径也在这棵树上进行。

从图3-2我们还可看到图中生成树的节点还比较少、节点间距离比较大，导致规划出的路径不够平滑、主要由长直的线段组成、有一些生硬的转折点，将RRT的最大连接距离（MaxConnectionDistance）从1.0改到0.5、0.25时，路径变得更加平滑，有利于机器人转向，但可能导致路径更加弯曲、路径长度变大（例如图3-3），效果反而不如PRM。

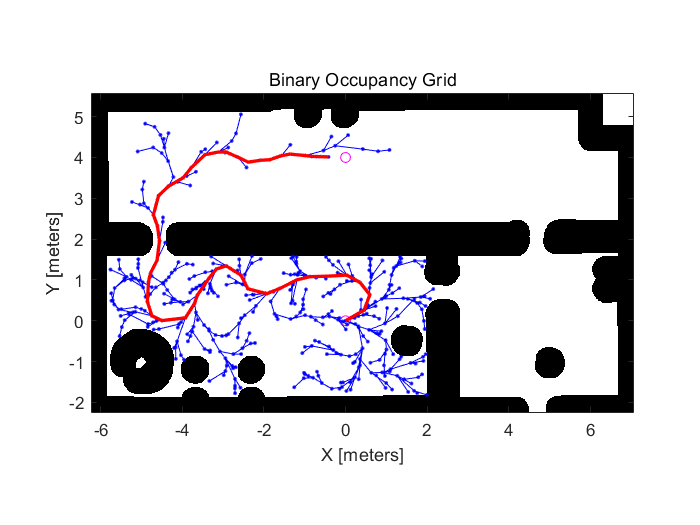


图 3- 3 RRT规划路径（MaxConnectionDistance=0.5）

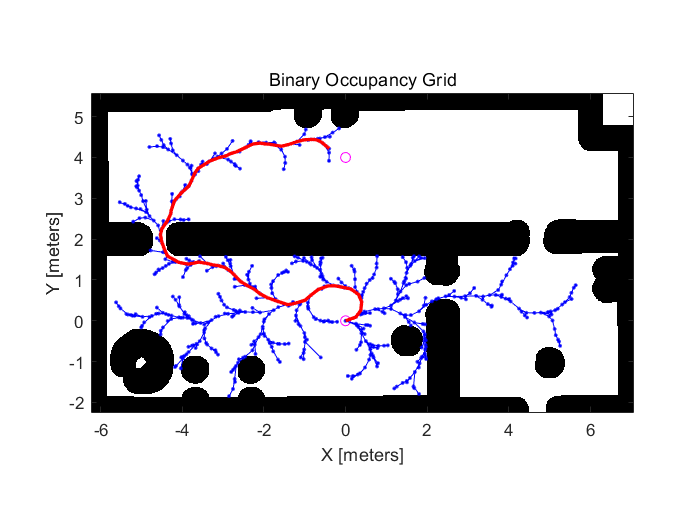


图 3- 4 RRT规划路径（MaxConnectionDistance=0.25）

# 总结

本次实习主要研究了基于 Matlab/Simulink 的 ROS 移动机器人运动控制问题。本次实习主要是为了让我们学习移动机器人路径规划和轨迹跟踪控制算法，以及在 Gazebo 环境下控制移动机器人，完成对其的运动控制。在实习中还学到了ROS 的基本原理以及ROS 的通讯机制，初步了解了如何使用虚拟机，在Gazebo 环境下进行移动机器人的运动控制仿真实验。

由于时间仓促，本次设计还有诸多不完善之处，例如路径规划中搜索最优路径步骤中，还可分别采用A\*和Dijstra算法并对比效果，在轨迹跟踪中，除了使用纯追踪控制器，还可使用Stanley的控制器。期望在今后学习中可以把这些方面做得更好、更完善。

总的来说，经过本次实习，收获还是很明显的，不仅巩固了大三学过的移动机器人路径规划知识，还锻炼了我独自解决问题和编程实践的能力。

最后，请容许我对XXX老师在本课程教学中的辛苦付出以及对学生的体谅包容表示感谢！

# 附录

1. clear;close all;clc;
2. %% 创建ROS环境
3. rosshutdown;
4. rosinit('192.168.153.130');
5. %% 路径规划
6. load("officemap.mat");
7. robotRadius=0.25;
8. mapInflated=copy(map);
9. %膨胀障碍物，得到机器人的位形空间
10. inflate(mapInflated,(robotRadius+0.05));
11. start = [0, 0, 0];%设置起点
12. goal = [0, 4, 0];%设置终点
14. bounds = [mapInflated.XWorldLimits; mapInflated.YWorldLimits; [-pi pi]];
15. ss = stateSpaceDubins(bounds);
16. ss.MinTurningRadius = 0.4; %最小转弯半径
17. stateValidator = validatorOccupancyMap(ss);
18. stateValidator.Map = mapInflated;
19. stateValidator.ValidationDistance = 0.05;
20. planner = plannerRRT(ss, stateValidator);
21. planner.MaxConnectionDistance = 0.25;
22. planner.MaxIterations = 30000;
23. rng(0,'twister')   %随机数生成
24. [pthObj, solnInfo] = plan(planner, start, goal); %使用RRT\*路径规划器规划车辆路径
25. %作图显示路径规划结果
26. figure ( );
27. show(mapInflated);%显示地图
28. hold on;
29. plot(start(1), start(2), 'mo',goal(1), goal(2), 'mo');%显示起点终点
30. hold on;
31. plot(solnInfo.TreeData(:,1), solnInfo.TreeData(:,2), 'b.-');%显示生成树
32. hold on;
33. plot(pthObj.States(:,1), pthObj.States(:,2), 'r-', 'LineWidth', 2)%显示路径
34. interpolate(pthObj,300) % 插值
35. % path=findpath(prm, startLocation, endLocation); %找出可行路径
36. % figure(4);
37. % show(prm); %作图显示路径规划结果
38. %% 定义机器人的当前位姿
39. robotCurrentLocation = [start(1)+0.02, start(2)-0.05];    %定义机器人的初始位置(设置误差)
40. robotGoal = [goal(1), goal(2)];                %定义机器人的目标位置
41. initialOrientation = 0+0.01;              %定义机器人的初始朝向(设置误差)
42. robotCurrentPose = [robotCurrentLocation initialOrientation];   %定义机器人的当前位姿
43. %% 创建控制器（纯追踪算法）
44. controller = controllerPurePursuit;      % 创建算法
45. controller.Waypoints = pthObj.States(:,1:2);             % 设置目标路径的组成点
46. controller.DesiredLinearVelocity = 0.3;
47. controller.MaxAngularVelocity = 3;
48. controller.LookaheadDistance = 0.5;
49. goalRadius=0.05;
50. distanceToGoal=norm(robotCurrentLocation - robotGoal);
51. %% 定义 Publisher 和 Subscriber
52. velcomd=rospublisher('/mobile\_base/commands/velocity');
53. %velcomd=rospublisher('/cmd\_vel');
54. vel=rosmessage(velcomd);
55. odomsub=rossubscriber('/odom');
56. %% 运动控制
57. while( distanceToGoal > goalRadius )
58. [v1,omega1] = step(controller, robotCurrentPose);          % 根据当前Pose，计算速度和转角
59. vel.Linear.X=v1;
60. vel.Angular.Z=omega1;
61. send(velcomd,vel);                                         % 驱动机器人
62. odom=receive(odomsub,1);                                   % 收集位姿信息
63. robotCurrentPose=MyreadPose(odom);
64. robotCurrentLocation = robotCurrentPose(1:2);              % 获取驱动后Pose
65. distanceToGoal = norm(robotCurrentLocation - robotGoal);   % 计算偏差
66. waitfor(rateControl(20));                                  % 延迟等待
67. end