****

智能自主机器人与系统

**轮式移动机器人规划与控制**

学院 (系)： 电子与信息工程学院

专 业： 自动化

学 号： 2150248

姓 名： 姚天亮

指导老师： 王祝萍、张皓

**摘要**

轮式移动机器人作为机器人的重要类型之一，其路径规划与运动控制问题具有重要意义。本项目通过采用A\*算法，采用轨迹跟踪控制，使用MATLAB完成算法实现与仿真验证，并进一步在ROS环境下集成开发，利用订阅-发布机制实现与移动机器人QBot2e的通信。实现了轮式移动机器人基于MATLAB与ROS的路径规划与运动控制，为机器人智能导航应用提供实验研究。

**关键词**：轮式移动机器人、规划、控制、ROS.

**智能自主机器人与系统**

轮式移动机器人规划与控制

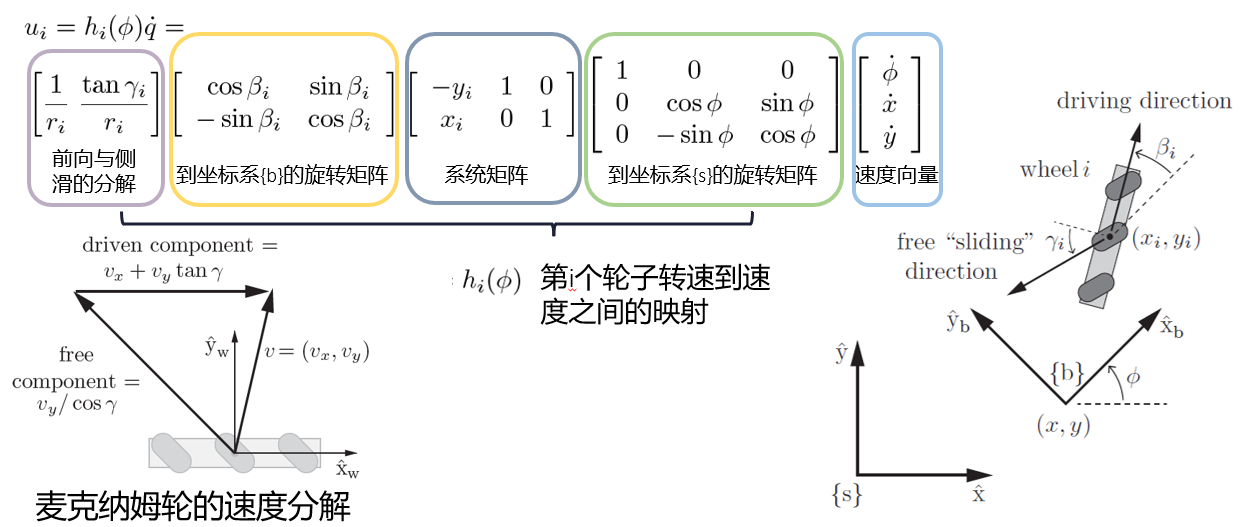
2150248-姚天亮-自动化

1. **轮式移动机器人：**

轮式机器人指的是一类主要以轮子旋转产生运动的机器人 [1]。

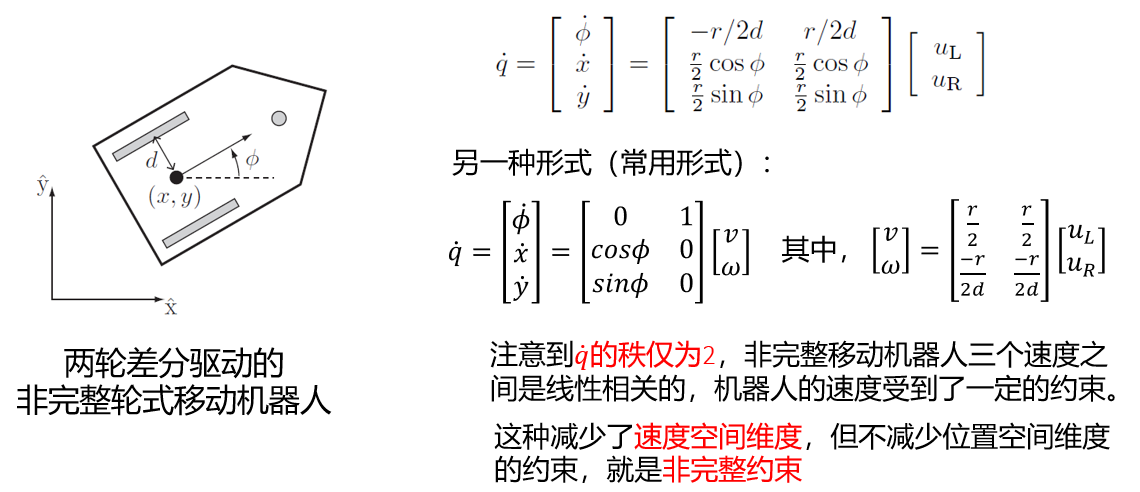
根据轮子种类的不同可以分为：全向移动（omnidirectional）轮式机器人以及非完整（nonholonomic）轮式移动机器人两大类。

全向移动轮式机器人的轮子通常是全向轮（Omniwheels）或麦克纳姆轮（Mecanum wheels），这两种轮子的特点是能够侧向滑动，因此机器人的速度不受到等式约束。



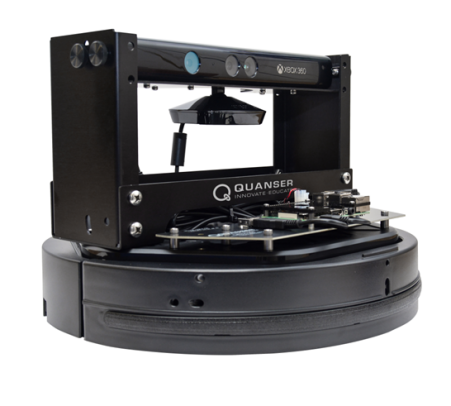
非完整移动轮式机器人的轮子则是标准轮，标准轮的特征就是不能侧向滑动，机器人的速度会受到以下等式约束：



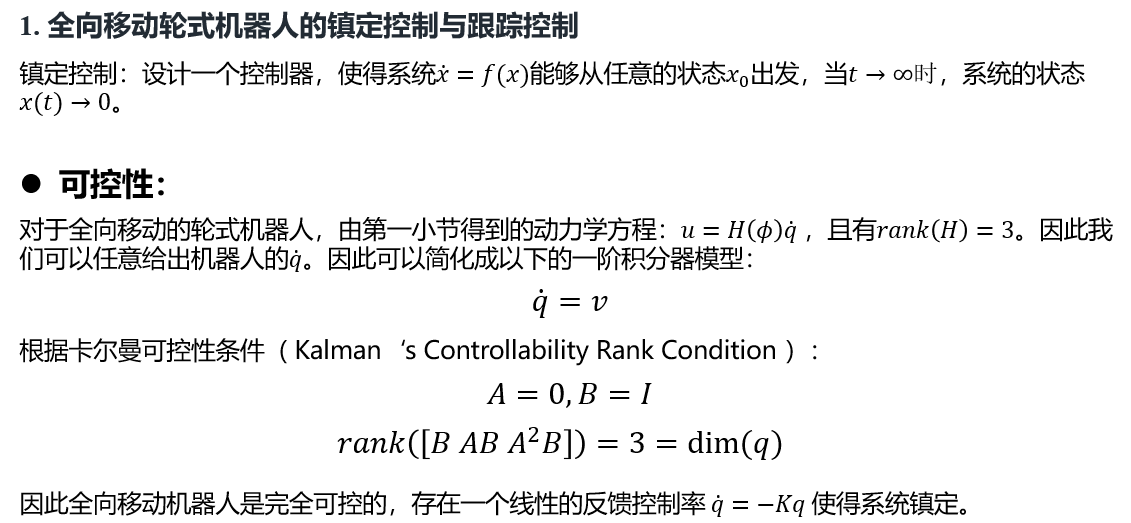


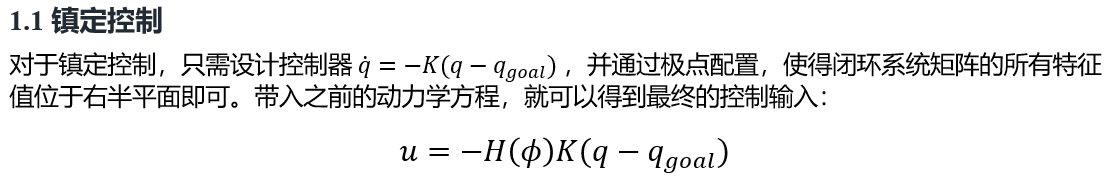
1. **问题描述：**

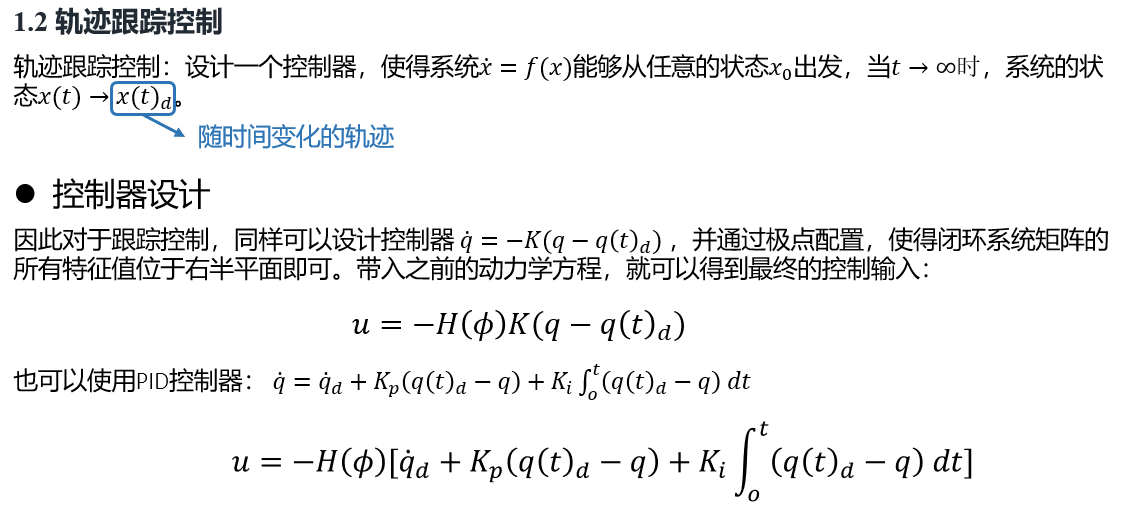
编写程序、建立动力学模型，实现对轮式移动机器人的闭环控制，并在给定的地面移动机器人 QBot 2e平台上，实现机器人沿指定路径（8字形）的路径跟随与轨迹镇定控制，使得机器人能够在满足位姿条件的情况下，自主沿着指定路径，到达指定位置。

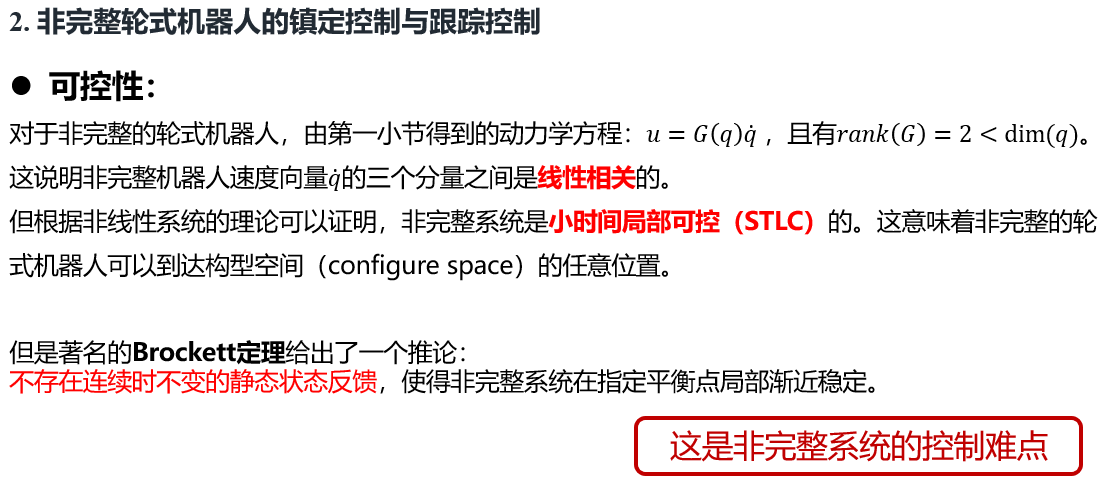


1. **轮式移动机器人的控制理论：**

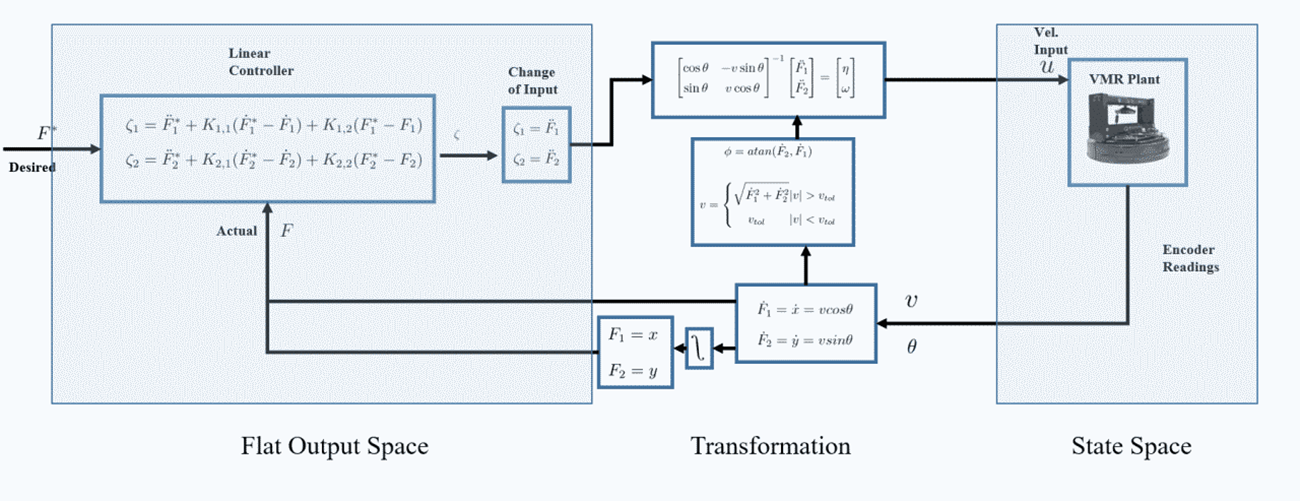


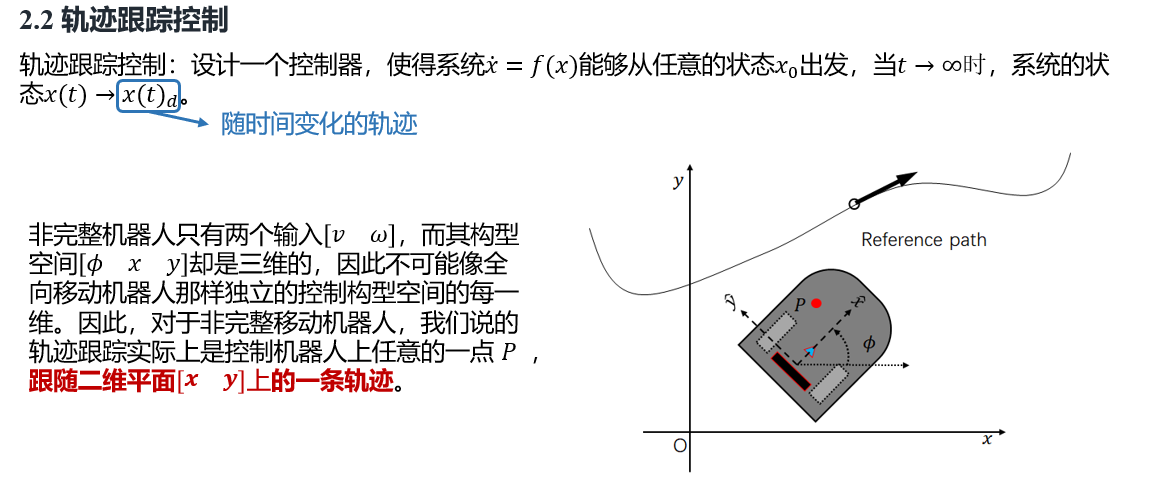


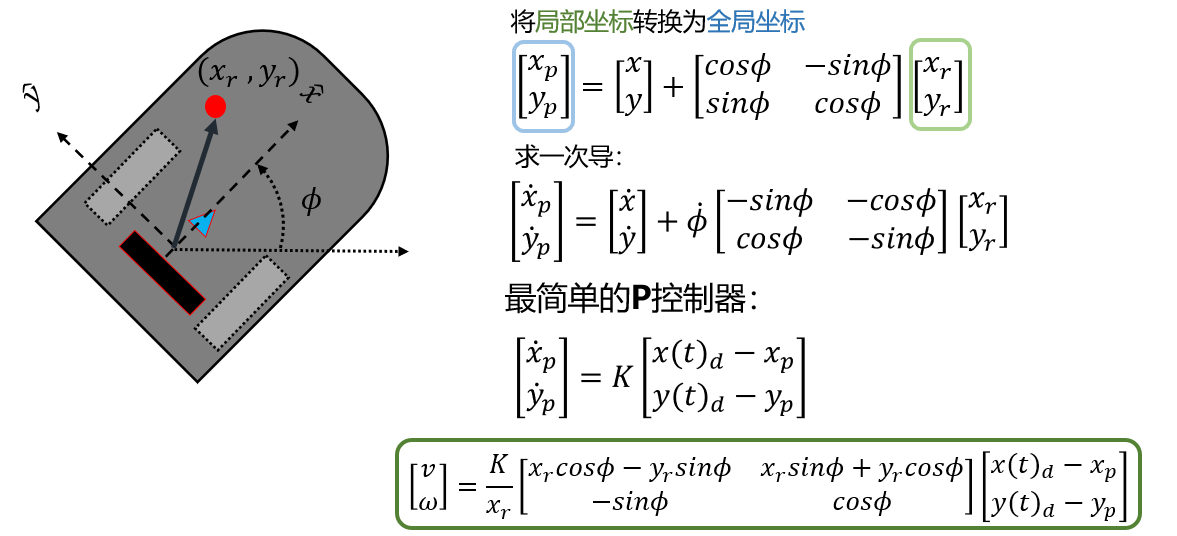




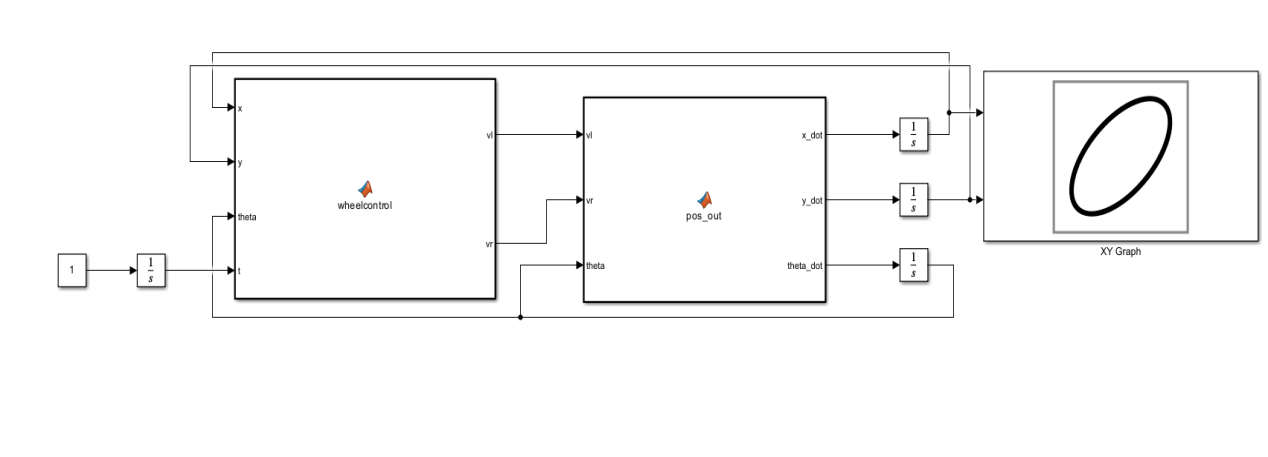
系统的最终实现框图如下：

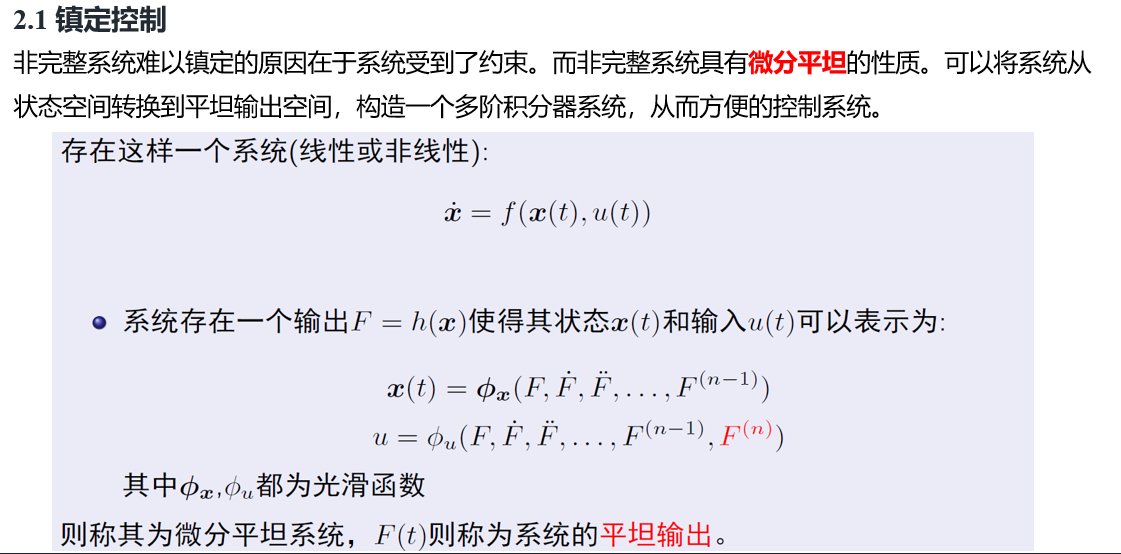
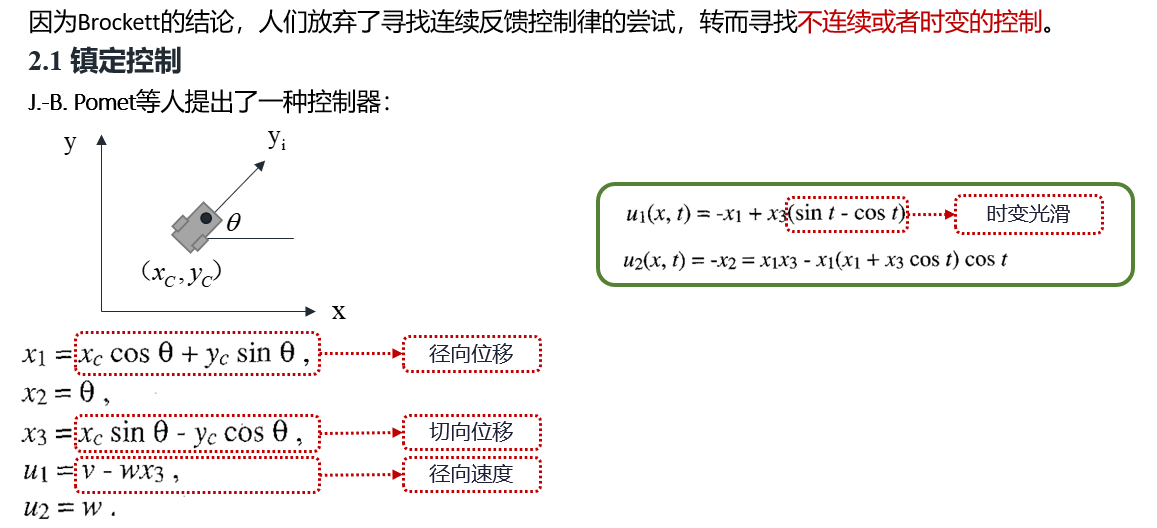






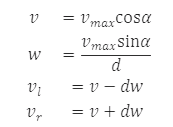
**四、控制器设计（轨迹跟随与镇定控制）**





**控制器设计：**

轨迹跟踪的公式是：



其中，v 是车辆的线速度，w 是车辆的角速度，vl​和 vr是左右车轮的速度，vmax 是车辆的最大速度，d 是车轮之间的距离，α 是车辆的方向与目标位置的连线的夹角 [2]。

在问题中，定态控制的给出的已知量有：

初始点 (x0, y0) 和终点 (x1,y1) 的坐标

终点车辆的朝向 θgoal​

车轮参数 d 和最大速度 vmax​

当前时间t

当前车辆的位置 (x,y) 和方向 θ

定态控制的目标是让车辆沿着一条二次函数轨迹从初始点运动到终点，并在终点保持给定的朝向。

为此，我们构造如下函数，以实现功能，具体函数及其作用如下：

1.pos\_out函数:

使用小车运动学模型,根据车轮线速度v，vr和方向角theta计算出小车的速度v和角速度w

根据v，w及theta利用坐标变换公式计算出小车在下一时刻的位置(x,y)和theta变化率theta\_dot

1. wheelcontrol函数轨迹跟踪部分:

计算当前位置(x,y,theta)与目标轨迹位置(x\_goal,y\_goal)的误差dx,dy

使用atan2函数根据误差计算小车需要采取的航向角alpha

alpha反转后与v\_max相乘得到小车需要的线速度v和角速度w

根据小车模型，从v,w计算出左右车轮的线速度vl,vr

wheelcontrol函数定点控制部分:

使用二次函数拟合得到不同时间下的目标位置(x\_goal,y\_goal)

计算当前位置与目标位置的误差dx，dy

如果误差大于设定值，同轨迹跟踪部分计算需要的v，w，进而计算vl，vr

如果误差在阈值内，以0速度静止在目标点

两个函数通过位置误差—速度—动力学模型的反馈闭环实现小车控制:

wheelcontrol根据误差计算速度指令

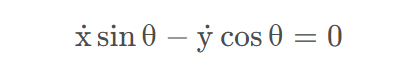
pos\_out根据速度指令计算小车动态状态

下一个循环再用更新后状态反馈到wheelcontrol

1、两轮式移动机器人参考点位于机器人驱动轴中心模型

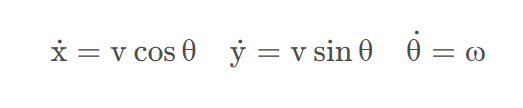
控制目标是使机器人在二维平面中运动到指定位置，但由于轮式机器人不能进行平移运动，故其状态变量不能只取（x,y），应考虑车头朝向问题，所以小车的状态变量用向量表示为：

（x,y）为机器人当前位置，theta为机器人的车头方向，机器人只能沿着垂直与驱动轴的方向移动，故移动机器人的运动约束为：



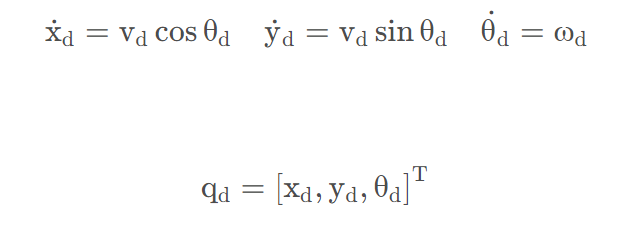
将机器人驱动轴方向的运动力抵消掉

在这个约束条件下，移动机器人的运动学模型可以用



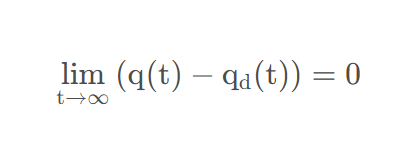
其中，v和omega分别表示移动机器人的线性速度和角速度

假设参考轨迹是可行的，并由以下虚拟机器人生成



2、Matlab仿真模型建立

那么我们的控制目标是为移动机器人设计一个连续反馈控制律(v，ω)，同时解决稳定和跟踪问题，控制目标为：

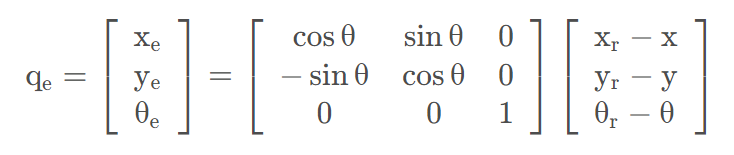


根据上述关系，我们最终要输出的为机器人的线速度与角速度，进而控制机器人跟踪期望轨迹，假定期望位置为

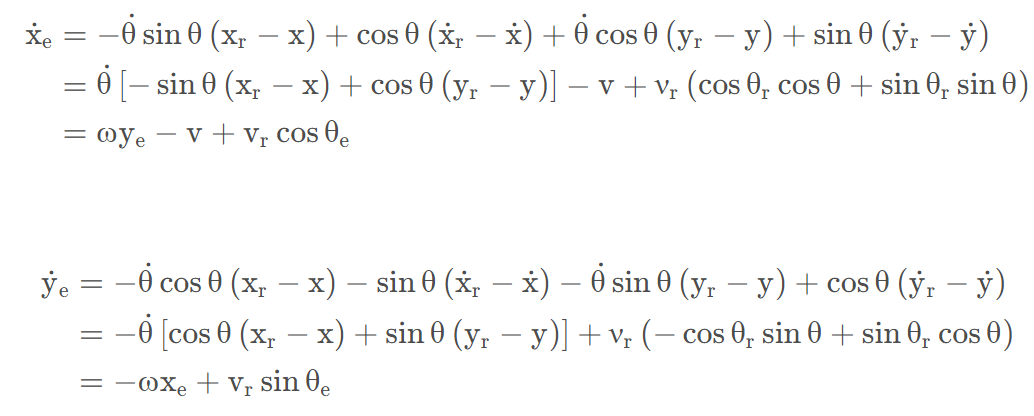
在x,y,θ方向上的误差分别为：



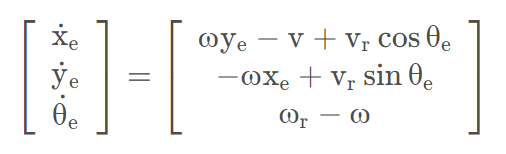
定义位姿跟踪误差：



求导得到：



整理位姿误差微分方程为：

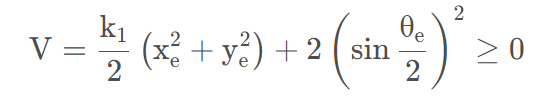


根据上述得到的运动学方程，自主导航车轨迹跟踪的问题转化为设计输入控制量

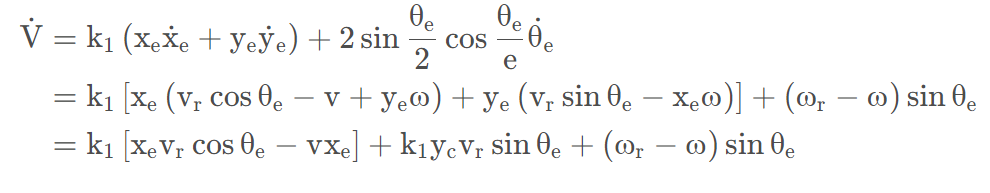
使闭环系统位姿误差 凡能够全局一致有界，并且当t → ∞，系统在任意的初始跟踪误差下有



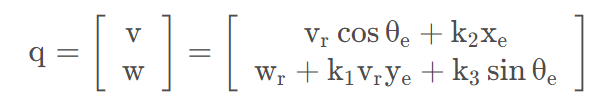
根据自主导航车的位姿误差微分方程，可以选取函数如下：



将其微分得：



由此可得控制律为：



**路径规划算法：**

众所周知，无人驾驶大致可以分为三个方面的工作：感知，决策及控制。

路径规划是感知和控制之间的决策阶段，主要目的是考虑到车辆动力学、机动能力以及相应规则和道路边界条件下，为车辆提供通往目的地的安全和无碰撞的路径。

路径规划问题可以分为两个方面：

（一）全局路径规划：全局路径规划算法属于静态规划算法，根据已有的地图信息（SLAM）为基础进行路径规划，寻找一条从起点到目标点的最优路径。

通常全局路径规划的实现包括Dijikstra算法，A\*算法，RRT算法等经典算法，也包括蚁群算法、遗传算法等智能算法；

（二）局部路径规划：局部路径规划属于动态规划算法，是无人驾驶汽车根据自身传感器感知周围环境，规划处一条车辆安全行驶所需的路线，常应用于超车，避障等情景。通常局部路径规划的实现包括动态窗口算法（DWA），人工势场算法，贝塞尔曲线算法等，也有学者提出神经网络等智能算法。

**A\*算法：**

实验要求我们对于A\_star\_search函数进行相应的填写，包含实现回溯的部分。即填写A\_star\_search中关于空白算法的部分。

A\*算法的基本思路为利用利用开启关闭列表来标定是否进行迭代循环判断过，每轮迭代循环判断从开启列表中选择一个最优解串联到关闭列表中，然后进行进一步地循环迭代计算[4]。最优解的选择与启发式函数的选择方式有关。选择出最优解后，把原来的拓展节点改入关闭列表，然后在拓展出的节点周围进行进一步的循环判断，并且将所有的周围一圈的点加入到开启列表中。这部分称为拓展部分。循环判断直至寻找到目标节点为止，然后退出循环，利用回溯算法进行对应路径的寻找。因为每一个拓展出来的节点必定拥有父节点，利用从最终节点一层层回溯回起点的算法称为回溯算法。这样最后反着写即为路径path。

下面给出A\*算法的伪代码：

输入：网格连通图，起点Node\_start，终点Node\_end

输出：联通起点和终点的路径

# 初始化

初始化Openlist为空，初始化图中所有的节点，也就是将所有属性置空。

将被障碍物占据的节点的is\_in\_Closedlist置为1

为初始节点完善相关属性：g = 0；利用曼哈顿距离计算h；parent\_idx = null ；is\_in\_Openlist = 0；is\_in\_Closedlist = 0

将初始节点加入Openlist，并将初始节点的is\_in\_Openlist置为1

# 循环while Openlist ！= 空 ：

从Openlist中选取一个代价f最小的节点，记为Node\_current

拓展Node\_current，将其所有子节点放置在临时集合A中

for each Node\_child in 集合A ：

if (Node\_child 超出地图边界)) or (Node\_child的is\_in\_Closedlist==1):

continue

if Node\_child的is\_in\_Openlist==0 :

计算Node\_child的历史代价g，预期代价h，总代价f

将Node\_child的父节点记为 parent\_idx = Node\_current

将Node\_child放入Openlist，并将Node\_child的is\_in\_Openlist置为1

else ：

计算将Node\_child的父节点换成Node\_current的话，总代价f\_alternative

if f\_alternative < Node\_child原有的f ：

将Node\_child的父节点更新为 parent\_idx = Node\_current

更新Node\_child的历史代价g和总代价f

将Node\_current从Openlist中移除。

将Node\_current对应属性改为：is\_in\_Openlist = 0；is\_in\_Closedlist = 1

if 集合A中包含终点Node\_end ：

break

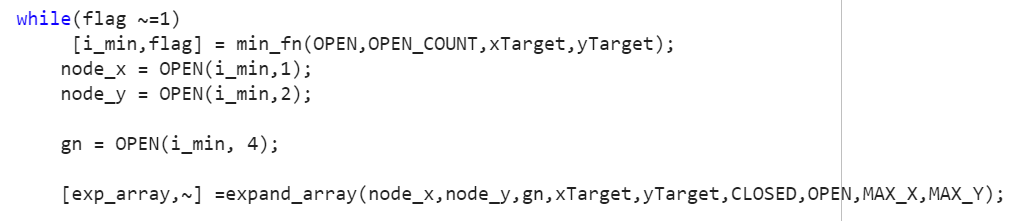
# 输出路径

构建空列表B，将Node\_end放入列表Bif 列表B中最后一个节点的父节点不是空 ：

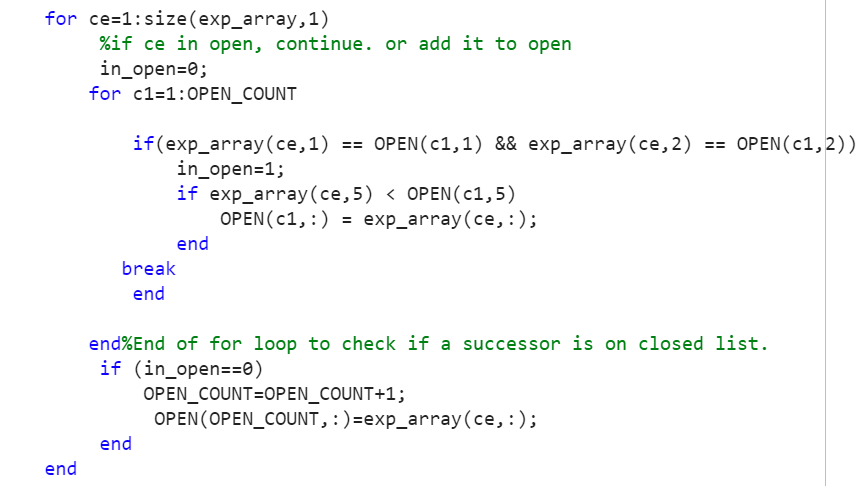
B.append(将列表B中最后一个节点的父节点)

将列表B逆序打印就是路径。

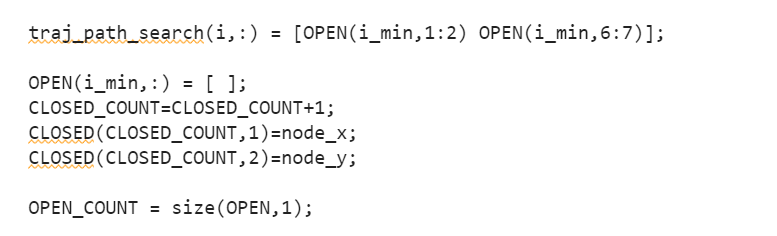
在程序中，我们按照伪代码的思路进行完善。



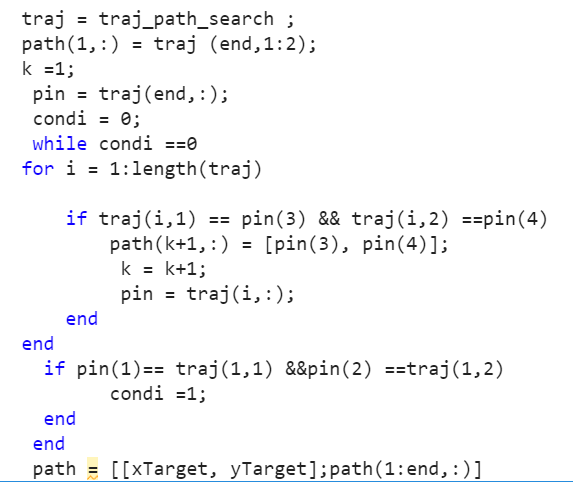
这个函数的功能是实现目标节点附近最小值节点的寻找，并且将寻找到的周围目标节点放入到开启数组中。i\_min负责进行最小值最优扩展节点的寻找，然后把找到的点的x、y坐标分开并且加入到open数组当中，然后根据最优扩展节点进行进一步的expand\_arrey扩展节点关于启发式节点的相关内容的计算。



扩展节点部分，根据已知Open数组以及障碍物Close数组进行进一步扩展结点之后的节点扩展。这个部分中间一段特别进行了新一步改进，比较了一下如果这次扩展出来的节点在之前已经扩展出来的节点中的情况，如果之前已经放在open数组里，那么进行一下启发式函数的相关比较，比较一下更新启发值更小的规划更优的路径。

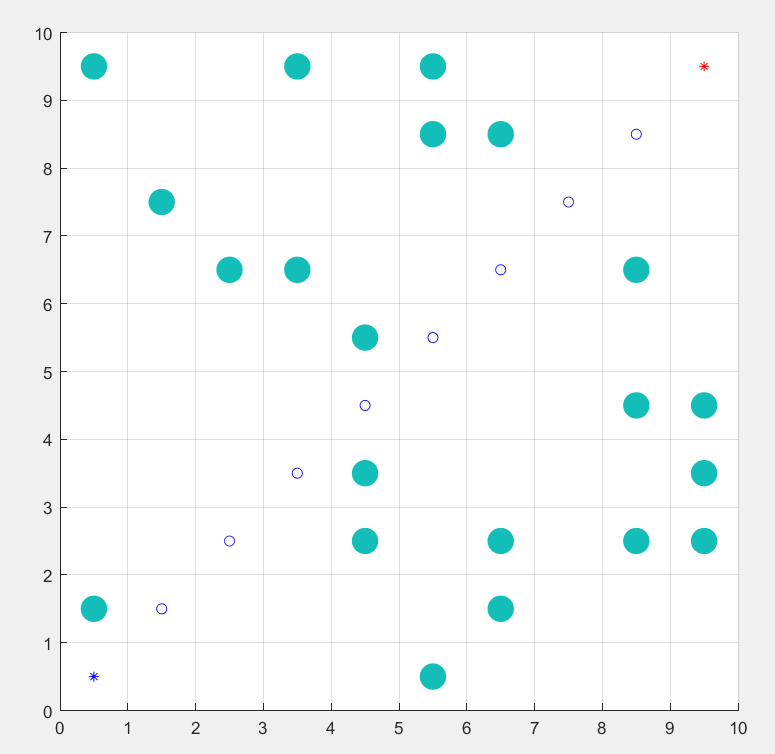


拓展结束后，将对应被拓展的节点加入到关闭列表中。



回溯算法进行路径的寻找。由于每个节点都是由之前的节点拓展而来，所以只要通过最后目标节点的回溯算法即可得出完整的路径信息。

在Matlab仿真中，基于A\*规划得到的路径如下



**RRT算法：**

RRT算法，即快速随机树算法（Rapid Random Tree），是LaValle在1998年首次提出的一种高效的路径规划算法。RRT算法以初始的一个根节点，通过随机采样的方法在空间搜索，然后添加一个又一个的叶节点来不断扩展随机树[5]。

当目标点进入随机树里面后，随机树扩展立即停止，此时能找到一条从起始点到目标点的路径。算法的计算过程如下：

step1：初始化随机树。将环境中起点作为随机树搜索的起点，此时树中只包含一个节点即根节点；

stpe2：在环境中随机采样。在环境中随机产生一个点，若该点不在障碍物范围内则计算随机树中所有节点到的欧式距离，并找到距离最近的节点，若在障碍物范围内则重新生成并重复该过程直至找到;

stpe3：生成新节点。在和连线方向，由指向固定生长距离生成一个新的节点，并判断该节点是否在障碍物范围内，若不在障碍物范围内则将添加到随机树 中，否则的话返回step2重新对环境进行随机采样；

step4：停止搜索。当和目标点之间的距离小于设定的阈值时，则代表随机树已经到达了目标点，将作为最后一个路径节点加入到随机树中，算法结束并得到所规划的路径 。

RRT算法由于其随机采样及概率完备性的特点，使得其具有如下优势：

（1）不需要对环境具体建模，有很强空间搜索能力；

（2）路径规划速度快；

（3）可以很好解决复杂环境下的路径规划问题。

但同样是因为随机性，RRT算法也存在很多不足的方面：

（1）随机性强，搜索没有目标性，冗余点多，且每次规划产生的路径都不一样，均不一是最优路径；

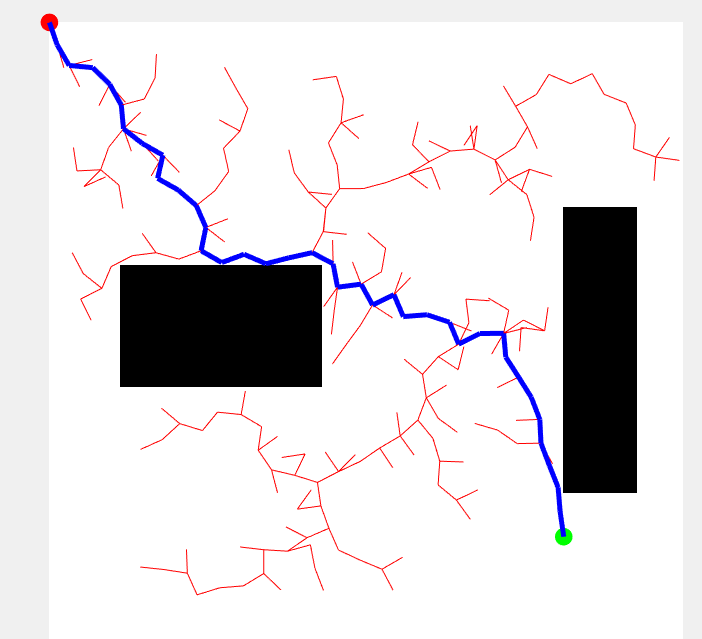
（2）可能出现计算复杂、所需的时间过长、易于陷入死区的问题；

（3）由于树的扩展是节点之间相连，使得最终生成的路径不平滑；

（4）不适合动态环境，当环境中出现动态障碍物时，RRT算法无法进行有效的检测；

（5）对于狭长地形，可能无法规划出路径。

在Matlab仿真中，基于RRT的路径规划结果如下：



## **五、具体实验**

### 1、QBOT 2E实验平台

Quanser QBot 2是一种创新的自主地面机器人系统，采用微软Kinect和树莓派嵌入式目标集成了坚固的教育用地面车辆。QBot 2e由Yujin Robot Kobuki平台、微软Kinect RGB摄像头和深度传感器以及无线嵌入式计算机(也称为目标计算机)组成。车载嵌入式计算机系统使用树莓派3 Model B+计算机运行Quanser实时控制软件QUARCr，并与QBot 2e数据采集卡(DAQ)接口。

### 2、代码编写平台

Matalb以及Simulink直接进行可编译程序的编写进而在实验平台上进行验证实验。这里选择使用Matlab进行实验的验证。

### 3、镇定控制和跟踪控制简介

（1）　镇定：控制机器人到达并稳定在某个静止的状态，实际生活中的例子就是把汽车停到一个指定的停车位里。  
（2）　跟踪：控制机器人跟随某个运动着的状态（即轨迹），实际生活中的例子就是让汽车沿着车道中心线行驶。

**4、ROS消息通信简介**

ROS（机器人操作系统，Robot Operating System），是专为机器人软件开发所设计出来的一套电脑操作系统架构。它是一个开源的元级操作系统（后操作系统），提供类似于操作系统的服务，包括硬件抽象描述、底层驱动程序管理、共用功能的执行、程序间消息传递、程序发行包管理，它也提供一些工具和库用于获取、建立、编写和执行多机融合的程序。本次实验利用ROS的话题通信功能，实现了基于自己的电脑（Master）与从机（slaver）进行通信。

1. **实验程序架构**

（1）机器人和地图参数初始化（如允许接受的最大角度和位置误差、允许接受的最大速度、轨迹半径、起点终点等）

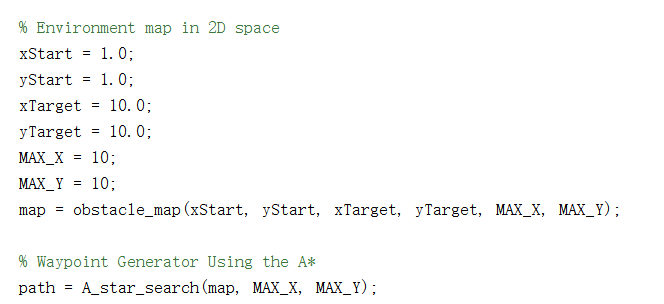
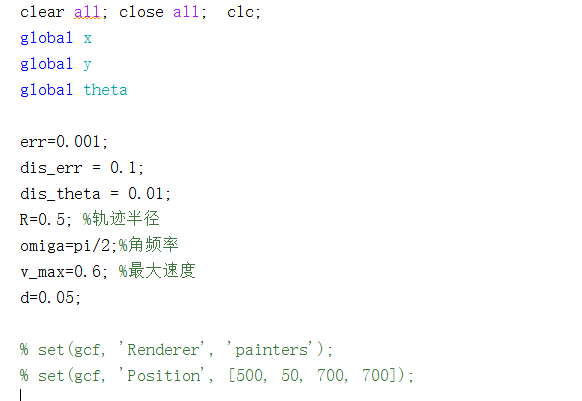
（2）记录机器人初始位置（这一步至关重要，在接下来的程序中，小组通过假设机器人初始位置为坐标原点建立地图）

（3）开始循环，在循环中，每次读取机器人当前的位置与目标节点（节点不一定是终点，可能是我们path中的某一点，其实就是让robot轮番经过path中的每个点）的距离，而后根据之前Q-bot基础实验中建立的机器人运动学模型（阿克曼模型），设定机器人车轮线速度和角速度，当迭代次数达到上限或者与终点的距离在误差可以接受的范围内的时候，让机器人停止运动。

（4）在Q-BOT基础实验上，本次小组在控制系统中加入PID闭环控制，提升了机器人运动的稳定性，让机器人在终点处的定态控制有更好的效果。

控制移动机器人路径移动代码，详细解说明如下:

第一部分是路径规划。本程序使用A\*算法(A\* algorithm)进行路径规划。A\*算法是一种广泛使用的启发式搜索算法,用于实现在有目标需求的环境中找到一条从起始位置到目标位置的最佳路径。它通过计算每个节点的F值来评估节点优先级,F值的计算考虑了从起点到当前节点的实际费用G值和从当前节点到目标的估计费用H值。通过迭代搜索并选择F值最小的节点扩展,最终可以找到最优路径。本代码使用地图信息和A\*算法实现路径规划。



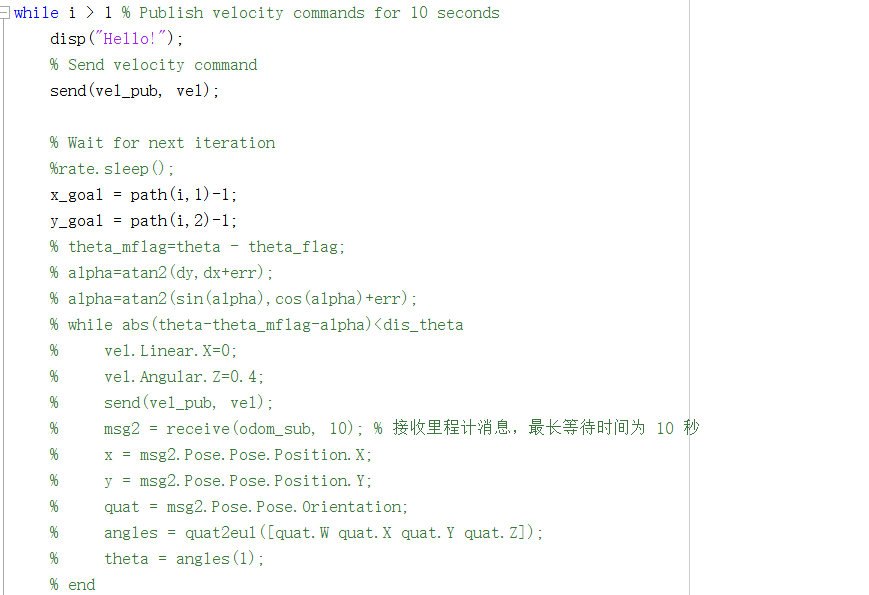
第二部分是ROS节点初始化。ROS(Robot Operating System)是一种通用机器人操作系统,本程序通过ROS实现机器人控制。首先初始化ROS节点,命名为“stabilization\_demo”,然后创建订阅机器人里程计信息的订阅对象odom\_sub,以及发布速度控制指令的发布对象vel\_pub。



第三部分是主循环控制。程序首先通过订阅里程计信息获取机器人当前实际位置信息x,y,theta。然后与路径规划点位置进行比较,计算出位置误差。根据位置误差与设定的阈值err进行判断,如果误差超过阈值,则需要进行运动控制。

第四部分是运动控制算法。程序根据位置误差计算出线速度vel.Linear.X和角速度vel.Angular.Z的控制信号。具体算法使用离心率控制法,根据位置误差α计算线速度和角速度的实时值。将控制信号通过ros发布对象vel\_pub发布给机器人。

第五部分是循环控制。主循环以10Hz的频率轮询控制机器人运动,知道路径规划结束,即所有点移动完毕。然后关闭ROS节点结束控制。





1. **实验结果**



在实验中Q-Bot沿着仿真得出的路径，在控制算法的控制下，通过ROS通信按照规定路线运行到指定位置，实验结果与现象符合预期。

## **六、心得体会**

在完成这个大作业后，我有以下心得体会：

路径规划算法对决定机器人行为非常重要。A\*和RRT都是经典的算法,各有优势,需要结合实际问题来选择。运动学模型的建立为控制奠定基础。ROS提供了很好的底层支持。但对初学者来说，配置有一些小复杂（感谢研究生学长的指导），还需多多实践。避障是机器人规划问题的一个重要方向，特别是我正在做的手术机器人规划[6]，通过简单仿真还是很难真实还原障碍物的影响，未来可以模拟多种情况来测试规划与控制策略，也可关注机器学习方法在机器人中的潜在应用。例如运用深度学习等技术进行环境建模与任务规划。多样化的感知技术有待研发。ROS可实现不同传感器的集成，但目前仅利用简单的定位信息。王老师在课上也给我们介绍了多模态传感的基本知识，将来可以尝试集成视觉等手段来提升功能。最后，团队协作也很重要，在过程中大家一起调试代码，交流沟通解决问题。

总体来说，这个项目帮助我系统学习了移动机器人的基础理论知识,也加强了代码设计和调试能力，还需不断学习和提高。

**参考文献**

1. Huq, R., Lacheray, H., Fulford, C., Wight, D., & Apkarian, J. (2009, May). QBOT: an educational mobile robot controlled in MATLAB Simulink environment. In 2009 Canadian Conference on Electrical and Computer Engineering (pp. 350-353). IEEE.
2. Lee, H., & Kim, H. J. (2017). Trajectory tracking control of multirotors from modelling to experiments: A survey. International Journal of Control, Automation and Systems, 15, 281-292.
3. Zhang, H. Y., Lin, W. M., & Chen, A. X. (2018). Path planning for the mobile robot: A review. Symmetry, 10(10), 450.
4. Liu, X., & Gong, D. (2011, April). A comparative study of A-star algorithms for search and rescue in perfect maze. In 2011 international conference on electric information and control engineering (pp. 24-27). IEEE.
5. Karaman, S., Walter, M. R., Perez, A., Frazzoli, E., & Teller, S. (2011, May). Anytime motion planning using the RRT. In 2011 IEEE international conference on robotics and automation (pp. 1478-1483). IEEE.
6. Yao, T., Wang, C., Wang, X., Li, X., Jiang, Z., & Qi, P. (2023). Enhancing percutaneous coronary intervention with heuristic path planning and deep-learning-based vascular segmentation. Computers in Biology and Medicine, 166, 107540.

**附件：核心代码**

1. **A\*算法**

# main.m

% Used for Motion Planning for Mobile Robots

% Thanks to HKUST ELEC 5660

close all; clear all; clc;

set(gcf, 'Renderer', 'painters');

set(gcf, 'Position', [500, 50, 700, 700]);

% Environment map in 2D space

xStart = 1.0;

yStart = 1.0;

xTarget = 10.0;

yTarget = 10.0;

MAX\_X = 10;

MAX\_Y = 10;

map = obstacle\_map(xStart, yStart, xTarget, yTarget, MAX\_X, MAX\_Y);

% Waypoint Generator Using the A\*

path = A\_star\_search(map, MAX\_X, MAX\_Y);

% visualize the 2D grid map

visualize\_map(map, path);

# A\_star\_search.m

function path = A\_star\_search(map, MAX\_X, MAX\_Y)

% This part is about map/obstacle/and other settings

% pre-process the grid map, add offset

size\_map = size(map, 1);

Y\_offset = 0;

X\_offset = 0;

% Define the 2D grid map array.

% Obstacle = -1, Target = 0, Start = 1

MAP = 2 \* (ones(MAX\_X, MAX\_Y));

% Initialize MAP with location of the target

xval = floor(map(size\_map, 1)) + X\_offset;

yval = floor(map(size\_map, 2)) + Y\_offset;

xTarget = xval;

yTarget = yval;

MAP(xval, yval) = 0;

% Initialize MAP with location of the obstacle

for i = 2: size\_map - 1

xval = floor(map(i, 1)) + X\_offset;

yval = floor(map(i, 2)) + Y\_offset;

MAP(xval, yval) = -1;

end

% Initialize MAP with location of the start point

xval = floor(map(1, 1)) + X\_offset;

yval = floor(map(1, 2)) + Y\_offset;

xStart = xval;

yStart = yval;

MAP(xStart, yStart) = 1;

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

% LISTS USED FOR ALGORITHM

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

% OPEN LIST STRUCTURE

%--------------------------------------------------------------------------

% IS ON LIST 1/0 | X val | Y val | Parent X val | Parent Y val | h(n) | g(n) | f(n) |

%--------------------------------------------------------------------------

OPEN = [];

% CLOSED LIST STRUCTURE

%--------------

% X val | Y val |

%--------------

CLOSED = [];

% Put all obstacles on the Closed list

k = 1; % Dummy counter

for i = 1: MAX\_X

for j = 1: MAX\_Y

if(MAP(i, j) == -1)

CLOSED(k, 1) = i;

CLOSED(k, 2) = j;

k = k + 1;

end

end

end

CLOSED\_COUNT = size(CLOSED, 1);

% set the starting node as the first node

xNode = xStart;

yNode = yStart;

OPEN\_COUNT = 1;

hn = distance(xNode, yNode, xTarget, yTarget);

gn = 0;

OPEN(OPEN\_COUNT, :) = insert\_open(xNode, yNode, xNode, yNode, hn, gn, hn + gn);

OPEN(OPEN\_COUNT, 1) = 0;

CLOSED\_COUNT = CLOSED\_COUNT + 1;

CLOSED(CLOSED\_COUNT, 1) = xNode;

CLOSED(CLOSED\_COUNT, 2) = yNode;

NoPath = 1;

%% This part is your homework

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

% START ALGORITHM

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

while(xNode ~= xTarget || yNode ~= yTarget) % you have to dicide the Conditions for while loop exit

exp\_array = expand\_array(xNode, yNode, gn, xTarget, yTarget, CLOSED, MAX\_X, MAX\_Y);

EXP\_COUNT = size(exp\_array,1);

for i= 1: 1: EXP\_COUNT

n\_node = exp\_array(i,:);

n\_xval = n\_node(1);

n\_yval= n\_node(2);

n\_hn = n\_node(3);

n\_gn = n\_node(4);

n\_fn = n\_node(5);

n\_index = node\_index(OPEN,n\_xval,n\_yval);

if n\_index > OPEN\_COUNT

OPEN\_COUNT = OPEN\_COUNT + 1;

OPEN(OPEN\_COUNT,:)= insert\_open(n\_xval, n\_yval,xNode, yNode, n\_hn, n\_gn, n\_fn);

else

if (OPEN(n\_index,1) == 1 && n\_fn < OPEN(n\_index,7))

OPEN(n\_index,4) = xNode;

OPEN(n\_index,5) = yNode;

OPEN(n\_index,6) = n\_hn;

OPEN(n\_index,7) = n\_gn;

OPEN(n\_index,8) = n\_fn;

end

end

end

i\_min = min\_fn(OPEN,OPEN\_COUNT,xTarget,yTarget);

if i\_min == -1

NoPath=0;

break;

end

OPEN(i\_min,1) = 0;

Node = OPEN(i\_min,:);

xNode = Node(2);

yNode = Node(3);

gn = Node(7);

end %end of while

% Once algorithm has run The optimal path is generated by starting of at the

% last node(if it is the target node) and then identifying its parent node

% until it reaches the start node.This is the optimal path

%

% How to get the optimal path after A\_star search?

% please finish it

%

path = [];

if NoPath == 1

xNode = xTarget;

yNode = yTarget;

PATH\_COUNT=1;

path(PATH\_COUNT,1)= xNode;

path(PATH\_COUNT,2)= yNode;

n\_index = node\_index(OPEN,xNode,yNode);

xParent = OPEN(n\_index,4);

yParent = OPEN(n\_index,5);

while(xParent ~= xNode || yParent ~= yNode)

xNode = xParent;

yNode = yParent;

PATH\_COUNT = PATH\_COUNT + 1;

path(PATH\_COUNT,1) = xNode;

path(PATH\_COUNT,2) = yNode;

n\_index = node\_index(OPEN,xNode,yNode);

xParent = OPEN(n\_index,4);

yParent = OPEN(n\_index,5);

end

end

end

# distance.m

function dist = distance(x1, y1, x2, y2)

% This function calculates the distance between any two cartesian

% coordinates.

% Copyright 2009-2010 The MathWorks, Inc.

%dist = sqrt((x1 - x2)^2 + (y1 - y2)^2);

dist=abs(x1-x2)+abs(y1-y2);

# expand\_array.m

function exp\_array=expand\_array(node\_x,node\_y,gn,xTarget,yTarget,CLOSED,MAX\_X,MAX\_Y)

%Function to return an expanded array

%This function takes a node and returns the expanded list

%of successors,with the calculated fn values.

%The criteria being none of the successors are on the CLOSED list.

%

%Copyright 2009-2010 The MathWorks, Inc.

%EXPANDED ARRAY FORMAT

%--------------------------------

%|X val |Y val ||h(n) |g(n)|f(n)|

%--------------------------------

exp\_array=[];

exp\_count=1;

c2=size(CLOSED,1);%Number of elements in CLOSED including the zeros

for k= 1:-1:-1

for j= 1:-1:-1

if (k~=j || k~=0) %The node itself is not its successor

s\_x = node\_x+k;

s\_y = node\_y+j;

if( (s\_x >0 && s\_x <=MAX\_X) && (s\_y >0 && s\_y <=MAX\_Y))%node within array bound

flag=1;

for c1=1:c2

if(s\_x == CLOSED(c1,1) && s\_y == CLOSED(c1,2))

flag=0;

end

end%End of for loop to check if a successor is on closed list.

if (flag == 1)

exp\_array(exp\_count,1) = s\_x;

exp\_array(exp\_count,2) = s\_y;

exp\_array(exp\_count,3) = distance(xTarget,yTarget,s\_x,s\_y);%distance between node and goal,hn

exp\_array(exp\_count,4) = gn+distance(node\_x,node\_y,s\_x,s\_y);%cost of travelling to node，gn

exp\_array(exp\_count,5) = exp\_array(exp\_count,3)+exp\_array(exp\_count,4);%fn

exp\_count=exp\_count+1;

end%Populate the exp\_array list!!!

end% End of node within array bound

end%End of if node is not its own successor loop

end%End of j for loop

end%End of k for loop

# insert\_open.m

function new\_row = insert\_open(xval, yval, parent\_xval, parent\_yval, hn, gn, fn)

% Function to Populate the OPEN LIST

% OPEN LIST FORMAT

%--------------------------------------------------------------------------

% IS ON LIST 1/0 | X val | Y val | Parent X val | Parent Y val | h(n) | g(n) | f(n) |

%-------------------------------------------------------------------------

%

% Copyright 2009-2010 The MathWorks, Inc.

new\_row = zeros(1,8);

new\_row(1, 1) = 1;

new\_row(1, 2) = xval;

new\_row(1, 3) = yval;

new\_row(1, 4) = parent\_xval;

new\_row(1, 5) = parent\_yval;

new\_row(1, 6) = hn;

new\_row(1, 7) = gn;

new\_row(1, 8) = fn;

end

# min\_fn.m

function i\_min = min\_fn(OPEN, OPEN\_COUNT, xTarget, yTarget)

% Function to return the Node with minimum fn

% This function takes the list OPEN as its input and returns the index of the

% node that has the least cost

%

% Copyright 2009-2010 The MathWorks, Inc.

temp\_array = [];

k = 1;

flag = 0;

goal\_index = 0;

for j = 1: 1: OPEN\_COUNT

if (OPEN(j, 1) == 1)

temp\_array(k, :) = [OPEN(j, :) j];

if (OPEN(j,2) == xTarget && OPEN(j,3) == yTarget)

flag = 1;

goal\_index = j; % Store the index of the goal node

end

k = k + 1;

end

end % Get all nodes that are on the list open

if flag == 1 % one of the successors is the goal node so send this node

i\_min = goal\_index;

end

% Send the index of the smallest node

if size(temp\_array) ~= 0

[~, temp\_min] = min(temp\_array(:, 8)); % Index of the smallest node in temp array

i\_min = temp\_array(temp\_min, 9); % Index of the smallest node in the OPEN array

else

i\_min = -1; % The temp\_array is empty i.e No more paths are available.

end

# node\_index.m

function n\_index = node\_index(OPEN, xval, yval)

% This function returns the index of the location of a node in the OPEN list

%

% Copyright 2009-2010 The MathWorks, Inc.

i = 1;

OPEN\_COUNT = size(OPEN, 1);

while(i <= OPEN\_COUNT && (OPEN(i, 2) ~= xval || OPEN(i,3) ~= yval))

i = i + 1;

end

n\_index = i;

end

# obstacle\_map.m

function map = obstacle\_map(xStart, yStart, xTarget, yTarget, MAX\_X, MAX\_Y)

% This function returns a map contains random distribution obstacles.

rand\_map = rand(MAX\_X, MAX\_Y);

map = [];

map(1, 1) = xStart;

map(1, 2) = yStart;

k = 2;

obstacle\_ratio = 0.25;

for i = 1: 1: MAX\_X

for j = 1: 1: MAX\_Y

if((rand\_map(i, j) < obstacle\_ratio) && (i ~= xStart || j ~= yStart) && (i ~= xTarget || j ~= yTarget))

map(k, 1) = i;

map(k, 2) = j;

k = k + 1;

end

end

end

map(k, 1) = xTarget;

map(k, 2) = yTarget;

end

# visualize\_map.m

function visualize\_map(map, path)

% This function visualizes the 2D grid map

% consist of obstacles/start point/target point/optimal path.

% obstacles

for obs\_cnt = 2: size(map, 1) - 1

scatter(map(obs\_cnt, 1) - 0.5, map(obs\_cnt, 2) - 0.5, 250, 155, 'filled');

hold on;

grid on;

% grid minor;

axis equal;

axis ([0 10 0 10]);

hold on;

end

% start point

scatter(map(1, 1) - 0.5, map(1, 2) - 0.5, 'blue', '\*');

hold on;

% target point

scatter(map(size(map, 1), 1) - 0.5, map(size(map, 1), 2) - 0.5, 'red', '\*');

hold on;

% optimal path

for path\_cnt = 2: size(path, 1) - 1

scatter(path(path\_cnt, 1) - 0.5, path(path\_cnt, 2) - 0.5, 'b');

hold on;

end

end

1. **RRT算法**

# collisionChecking.m

function feasible=collisionChecking(startPose,goalPose,map)

feasible=true;

dir=atan2(goalPose(1)-startPose(1),goalPose(2)-startPose(2));

for r=0:0.5:sqrt(sum((startPose-goalPose).^2))

posCheck = startPose + r.\*[sin(dir) cos(dir)];

if ~(feasiblePoint(ceil(posCheck),map) && feasiblePoint(floor(posCheck),map) && ...

feasiblePoint([ceil(posCheck(1)) floor(posCheck(2))],map) && feasiblePoint([floor(posCheck(1)) ceil(posCheck(2))],map))

feasible=false;break;

end

if ~feasiblePoint([floor(goalPose(1)),ceil(goalPose(2))],map), feasible=false; end

end

function feasible=feasiblePoint(point,map)

feasible=true;

if ~(point(1)>=1 && point(1)<=size(map,1) && point(2)>=1 && point(2)<=size(map,2) && map(point(2),point(1))==255)

feasible=false;

End

# RRT.m

%% 流程初始化

clc

clear all; close all;

x\_I=1; y\_I=1; % 设置初始点

x\_G=650; y\_G=650; % 设置目标点（可尝试修改终点）

Thr=50; % 设置目标点阈值，表示当前节点到达目标点方圆50内，就算作到达目标点

Delta= 30; % 设置扩展步长

x\_goal = [x\_G y\_G];

%% 建树初始化

T.v(1).x = x\_I; % T是我们要做的树，v是节点，这里先把起始点加入到T里面来

T.v(1).y = y\_I;

T.v(1).xPrev = x\_I; % 起始节点的父节点仍然是其本身

T.v(1).yPrev = y\_I;

T.v(1).dist=0; % 从父节点到该节点的距离，这里可取欧氏距离

T.v(1).indPrev = 0; %

%% 开始构建树，作业部分

% getDistance

getDist = @(x1,y1,x2,y2) sqrt((x1-x2)^2+(y1-y2)^2);

% end of getDistance

figure(1);

ImpRgb=imread('newmap.png');

Imp=rgb2gray(ImpRgb);

imshow(Imp)

xL=size(Imp,2);%地图x轴长度

yL=size(Imp,1);%地图y轴长度

hold on

plot(x\_I, y\_I, 'ro', 'MarkerSize',10, 'MarkerFaceColor','r');

plot(x\_G, y\_G, 'go', 'MarkerSize',10, 'MarkerFaceColor','g');% 绘制起点和目标点

count=1;

bFind = false;

tic

for iter = 1:30000

%Step 1: 在地图中随机采样一个点x\_rand，这个点为图中的坐标

%提示：用（x\_rand(1),x\_rand(2)）表示环境中采样点的坐标

x\_rand=[0 + (xL - 0) \* rand(1,1), 0 + (yL - 0) \* rand(1,1) ];

%Step 2: 遍历树，从树中找到最近邻近点x\_near

%提示：x\_near已经在树T里

distArr = arrayfun(@(dot) getDist(x\_rand(1),x\_rand(2),dot.x,dot.y),T.v);

[~,index] = min(distArr);

x\_near=[T.v(index).x,T.v(index).y];

%Step 3: 扩展得到x\_new节点

%提示：注意使用扩展步长Delta

theta = atan2(x\_rand(2)-x\_near(2),x\_rand(1)-x\_near(1));

x\_new=[x\_near(1)+Delta\*cos(theta), x\_near(2) + Delta\*sin(theta)];

%检查节点是否是collision-free

if ~collisionChecking(x\_near,x\_new,Imp)

continue;

end

count=count+1;

%Step 4: 将x\_new插入树T

%提示：新节点x\_new的父节点是x\_near

T.v(count).x = x\_new(1);

T.v(count).y = x\_new(2);

T.v(count).xPrev = x\_near(1);

T.v(count).yPrev = x\_near(2);

T.v(count).dist=Delta;

T.v(count).indPrev = index;

%Step 5:检查是否到达目标点附近

%提示：注意使用目标点阈值Thr，若当前节点和终点的欧式距离小于Thr，则跳出当前for循环

if getDist(x\_new(1),x\_new(2),x\_G,y\_G) < Thr

plot([x\_near(1),x\_new(1)],[x\_near(2),x\_new(2)],'r');

hold on;

bFind = true;

break;

end

%Step 6:将x\_near和x\_new之间的路径画出来

%提示 1：使用plot绘制，因为要多次在同一张图上绘制线段，所以每次使用plot后需要接上hold on命令

%提示 2：在判断终点条件弹出for循环前，记得把x\_near和x\_new之间的路径画出来

plot([x\_near(1),x\_new(1)],[x\_near(2),x\_new(2)],'r');

hold on;

pause(0.05); %暂停一会，使得RRT扩展过程容易观察

end

toc

%% 路径已经找到，反向查询

if bFind

path.pos(1).x = x\_G; path.pos(1).y = y\_G;

path.pos(2).x = T.v(end).x; path.pos(2).y = T.v(end).y;

pathIndex = T.v(end).indPrev; % 终点加入路径

j=0;

while 1

path.pos(j+3).x = T.v(pathIndex).x;

path.pos(j+3).y = T.v(pathIndex).y;

pathIndex = T.v(pathIndex).indPrev;

if pathIndex == 1

break

end

j=j+1;

end % 沿终点回溯到起点

path.pos(end+1).x = x\_I; path.pos(end).y = y\_I; % 起点加入路径

for j = 2:length(path.pos)

plot([path.pos(j).x; path.pos(j-1).x;], [path.pos(j).y; path.pos(j-1).y], 'b', 'Linewidth', 3);

end

else

disp('Error, no path found!');

end

1. **机器人运动控制**

# stabilization.m

clear all; close all; clc;

global x

global y

global theta

err=0.001;

dis\_err = 0.1;

dis\_theta = 0.01;

R=0.5; %轨迹半径

omiga=pi/2;%角频率

v\_max=0.6; %最大速度

d=0.05;

% set(gcf, 'Renderer', 'painters');

% set(gcf, 'Position', [500, 50, 700, 700]);

% Environment map in 2D space

xStart = 1.0;

yStart = 1.0;

xTarget = 10.0;

yTarget = 10.0;

MAX\_X = 10;

MAX\_Y = 10;

map = obstacle\_map(xStart, yStart, xTarget, yTarget, MAX\_X, MAX\_Y);

% Waypoint Generator Using the A\*

path = A\_star\_search(map, MAX\_X, MAX\_Y);

% clear; close all; clc;

%setenv('ROS\_MASTER\_URI','http://localhost:11311'); % Replace with actual ROS\_MASTER\_URI

setenv('ROS\_IP','10.1.1.104'); % Replace with your IP address

% Initialize ROS node

node\_name = 'stabilization\_demo\_233333';

setenv('ROS\_MASTER\_URI', 'http://10.1.1.4:11311')

rosinit('NodeName', node\_name);

x=0;

y=0;

theta=0;

% Create subscriber for odom topic

odom\_sub = rossubscriber('/Qbot2e\_121/odom', 'DataFormat', 'struct');

msg2 = receive(odom\_sub, 10); % 接收里程计消息，最长等待时间为 10 秒

x = msg2.Pose.Pose.Position.X;

y = msg2.Pose.Pose.Position.Y;

quat = msg2.Pose.Pose.Orientation;

angles = quat2eul([quat.W quat.X quat.Y quat.Z]);

theta = angles(1);

x\_flag = x;

y\_flag=y;

theta\_flag=theta;

% Create publisher for velocity commands

vel\_pub = rospublisher('/Qbot2e\_121/mobile\_base/commands/velocity', 'DataFormat', 'struct');

vel = rosmessage(vel\_pub);

% Set desired velocity values

% vel.Linear.X = 0.1; % Forward velocity

% vel.Angular.Z = 0.5; % Angular velocity

rate = robotics.Rate(10); % Rate of publishing velocity commands (10 Hz)

[i,le]=size(path);

while i > 1 % Publish velocity commands for 10 seconds

disp("Hello!");

% Send velocity command

send(vel\_pub, vel);

% Wait for next iteration

%rate.sleep();

x\_goal = path(i,1)-1;

y\_goal = path(i,2)-1;

% theta\_mflag=theta - theta\_flag;

% alpha=atan2(dy,dx+err);

% alpha=atan2(sin(alpha),cos(alpha)+err);

% while abs(theta-theta\_mflag-alpha)<dis\_theta

% vel.Linear.X=0;

% vel.Angular.Z=0.4;

% send(vel\_pub, vel);

% msg2 = receive(odom\_sub, 10); % 接收里程计消息，最长等待时间为 10 秒

% x = msg2.Pose.Pose.Position.X;

% y = msg2.Pose.Pose.Position.Y;

% quat = msg2.Pose.Pose.Orientation;

% angles = quat2eul([quat.W quat.X quat.Y quat.Z]);

% theta = angles(1);

% end

while abs(x-x\_flag - x\_goal) > dis\_err || abs(y-y\_flag - y\_goal) > dis\_err

x-x\_flag

y-y\_flag

theta-theta\_flag

dx=x\_goal-x+x\_flag;

dy=y\_goal-y+y\_flag;

alpha=atan2(dy,dx+err)-theta;

alpha=atan2(sin(alpha),cos(alpha)+err);

% vel.Linear.X=0.1;

% vel.Angular.Z=0.1;

vel.Linear.X = v\_max\*cos(alpha);

vel.Angular.Z = v\_max\*sin(alpha)/d;

% vel.Linear.X=0.6;

% vel.Angular.Z=0;

send(vel\_pub, vel);

msg2 = receive(odom\_sub, 10); % 接收里程计消息，最长等待时间为 10 秒

x = msg2.Pose.Pose.Position.X;

y = msg2.Pose.Pose.Position.Y;

quat = msg2.Pose.Pose.Orientation;

angles = quat2eul([quat.W quat.X quat.Y quat.Z]);

theta = angles(1);

end

i=i-1;

end

% Stop the robot by sending zero velocity command

% vel.Linear.X = 0;

% vel.Angular.Z = 0;

% send(vel\_pub, vel);

% Shutdown ROS

rosshutdown;

% Callback function for odom topic subscriber

% function odomCallback(src, msg)

% % Process odom data

% pose = msg.Pose.Pose;

% position = pose.Position;

% orientation = pose.Orientation;

%

% disp(['Position: X=',num2str(position.X),' Y=',num2str(position.Y),' Z=',num2str(position.Z)]);

% disp(['Orientation: X=',num2str(orientation.X),' Y=',num2str(orientation.Y),' Z=',num2str(orientation.Z),' W=',num2str(orientation.W)]);

% end

1. **ROS配置**

# Text.m

function main()

clear all; close all; clc;

% 设置 ROS 环境

setenv('ROS\_IP','10.1.1.104');

setenv('ROS\_MASTER\_URI', 'http://10.1.1.4:11311');

rosinit('NodeName', 'stabilization\_demo\_2333');

% 获取起点、终点和地图信息

xStart = 1.0; yStart = 1.0;

xTarget = 10.0; yTarget = 10.0;

MAX\_X = 10; MAX\_Y = 10;

map = obstacle\_map(xStart, yStart, xTarget, yTarget, MAX\_X, MAX\_Y);

path = A\_star\_search(map, MAX\_X, MAX\_Y);

% 初始化机器人位置和速度发布器

[odom\_sub, vel\_pub, x, y, theta] = initRobot();

% 设置控制参数

v\_max = 0.6;

d = 0.05;

err = 0.001;

dis\_err = 0.01;

% 控制循环

[i, ~] = size(path);

while i > 1

disp("Hello!");

% 更新机器人位置

[x, y, theta] = updateRobotPose(odom\_sub, x, y, theta);

% 计算目标位置和角度

x\_goal = path(i, 1) - 1;

y\_goal = path(i, 2) - 1;

% 控制机器人移动

controlRobot(x, y, theta, x\_goal, y\_goal, v\_max, d, err, dis\_err, vel\_pub, odom\_sub);

% 更新迭代

i = i - 1;

end

% 关闭 ROS

rosshutdown;

end

function [odom\_sub, vel\_pub, x, y, theta] = initRobot()

% 订阅 odom 主题

odom\_sub = rossubscriber('/Qbot2e\_121/odom', 'DataFormat', 'struct');

msg = receive(odom\_sub, 10);

x = msg.Pose.Pose.Position.X;

y = msg.Pose.Pose.Position.Y;

% 创建速度发布器

vel\_pub = rospublisher('/Qbot2e\_121/mobile\_base/commands/velocity', 'DataFormat', 'struct');

end

function [x, y, theta] = updateRobotPose(odom\_sub, x, y, theta)

% 接收 odom 数据

msg = receive(odom\_sub, 10);

x = msg.Pose.Pose.Position.X;

y = msg.Pose.Pose.Position.Y;

quat = msg.Pose.Pose.Orientation;

angles = quat2eul([quat.W quat.X quat.Y quat.Z]);

theta = angles(1);

end

function controlRobot(x, y, theta, x\_goal, y\_goal, v\_max, d, err, dis\_err, vel\_pub, odom\_sub)

while abs(x - x\_goal) > dis\_err && abs(y - y\_goal) > dis\_err

dx = x\_goal - x;

dy = y\_goal - y;

alpha = atan2(dy, dx + err) - theta;

alpha = atan2(sin(alpha), cos(alpha) + err);

% 计算线速度和角速度

linear\_vel = v\_max \* cos(alpha);

angular\_vel = v\_max \* sin(alpha) / d;

% 发布速度命令

sendVelocityCommand(vel\_pub, linear\_vel, angular\_vel);

% 更新机器人位置

[x, y, theta] = updateRobotPose(odom\_sub, x, y, theta);

end

end

function sendVelocityCommand(vel\_pub, linear\_vel, angular\_vel)

% 发布速度命令

vel = rosmessage(vel\_pub);

vel.Linear.X = linear\_vel;

vel.Angular.Z = angular\_vel;

send(vel\_pub, vel);

end