同濟大學
TONGJI UNIVERSITY

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

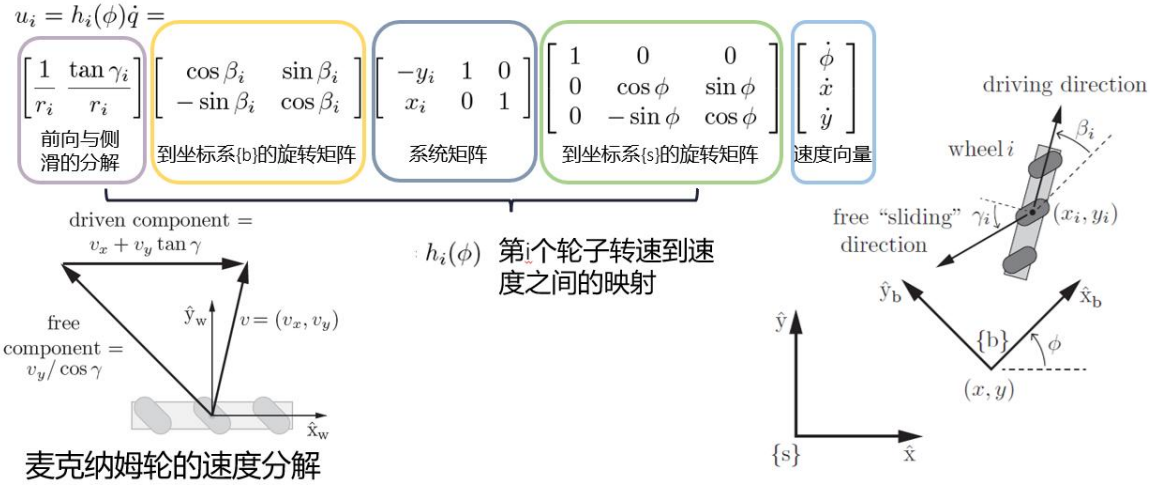
2150248-姚天亮-自动化

一、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]。

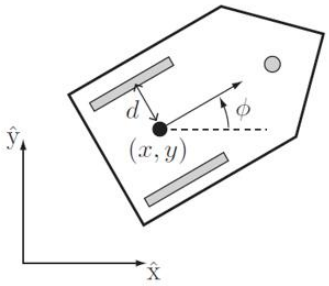
根据轮子种类的不同可以分为：全向移动（omnidirectional）轮式机器人以及非完整（nonholonomic）轮式移动机器人两大类。

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：

$$\dot{x}_c \sin\theta - \dot{y}_c \cos\theta = 0$$



两轮差分驱动的
非完整轮式移动机器人

$$\dot{q} = \begin{bmatrix} \dot{\phi} \\ \dot{x} \\ \dot{y} \end{bmatrix} = \begin{bmatrix} -r/2d & r/2d \\ \frac{r}{2}\cos\phi & \frac{r}{2}\cos\phi \\ \frac{r}{2}\sin\phi & \frac{r}{2}\sin\phi \end{bmatrix} \begin{bmatrix} u_{\mathrm{L}} \\ u_{\mathrm{R}} \end{bmatrix}$$

另一种形式（常用形式）：

$$\dot{q} = \begin{bmatrix} \dot{\phi} \\ \dot{x} \\ \dot{y} \end{bmatrix} = \begin{bmatrix} 0 & 1 \\ cos\phi & 0 \\ sin\phi & 0 \end{bmatrix} \begin{bmatrix} v \\ \omega \end{bmatrix} \quad \text{其中，} \begin{bmatrix} v \\ \omega \end{bmatrix} = \begin{bmatrix} \frac{r}{2} & \frac{r}{2} \\ \frac{-r}{2d} & \frac{-r}{2d} \end{bmatrix} \begin{bmatrix} u_L \\ u_R \end{bmatrix}$$

注意到$\dot{q}$的秩仅为2，非完整移动机器人三个速度之间是线性相关的，机器人的速度受到了一定的约束。

这种减少了速度空间维度，但不减少位置空间维度的约束，就是非完整约束

二、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。

三、轮式移动机器人的控制理论：

1. 全向移动轮式机器人的镇定控制与跟踪控制

镇定控制：设计一个控制器，使得系统$\dot{x} = f(x)$能够从任意的状态x_0出发，当$t \to \infty$时，系统的状态$x(t) \to 0$。

● 可控性：

对于全向移动的轮式机器人，由第一小节得到的动力学方程：$u = H(\phi)\dot{q}$，且有$rank(H) = 3$。因此我们可以任意给出机器人的$\dot{q}$。因此可以简化成以下的一阶积分器模型：

$$\dot{q} = v$$

根据卡尔曼可控性条件（Kalman ‘s Controllability Rank Condition）：

$$A = 0, B = I$$

$$rank([B\ AB\ A^2B]) = 3 = \dim(q)$$

因此全向移动机器人是完全可控的，存在一个线性的反馈控制率$\dot{q} = -Kq$使得系统镇定。

1.1 镇定控制

对于镇定控制，只需设计控制器 $\dot{q} = -K(q - q_{goal})$，并通过极点配置，使得闭环系统矩阵的所有特征值位于右半平面即可。带入之前的动力学方程，就可以得到最终的控制输入：

$$u = -H(\phi)K(q - q_{goal})$$

1.2 轨迹跟踪控制

轨迹跟踪控制：设计一个控制器，使得系统$\dot{x} = f(x)$能够从任意的状态x_0出发，当$t \to \infty$时，系统的状态$x(t) \to x(t)_d$。

随时间变化的轨迹

- 控制器设计

因此对于跟踪控制，同样可以设计控制器 $\dot{q} = -K(q - q(t)_d)$，并通过极点配置，使得闭环系统矩阵的所有特征值位于右半平面即可。带入之前的动力学方程，就可以得到最终的控制输入：

$$u = -H(\phi)K(q - q(t)_d)$$

也可以使用PID控制器：$\dot{q} = \dot{q}_d + K_p(q(t)_d - q) + K_i \int_o^t (q(t)_d - q)\, dt$

$$u = -H(\phi)[\dot{q}_d + K_p(q(t)_d - q) + K_i \int_o^t (q(t)_d - q)\, dt]$$

2. 非完整轮式机器人的镇定控制与跟踪控制

- **可控性：**

对于非完整的轮式机器人，由第一小节得到的动力学方程：$u = G(q)\dot{q}$，且有$rank(G) = 2 < \dim(q)$。

这说明非完整机器人速度向量$\dot{q}$的三个分量之间是**线性相关**的。

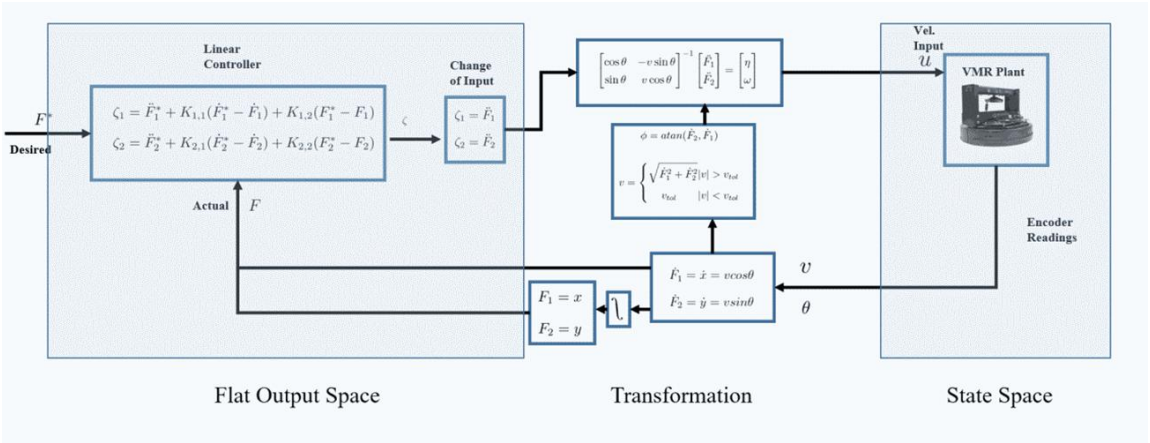
但根据非线性系统的理论可以证明，非完整系统是**小时间局部可控（STLC）**的。这意味着非完整的轮式机器人可以到达构型空间（configure space）的任意位置。

但是著名的**Brockett定理**给出了一个推论：

不存在连续时不变的静态状态反馈，使得非完整系统在指定平衡点局部渐近稳定。

这是非完整系统的控制难点

系统的最终实现框图如下：

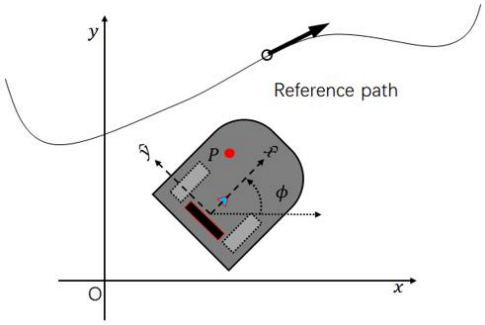


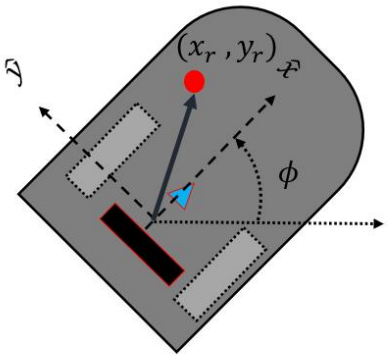
2.2 轨迹跟踪控制

轨迹跟踪控制：设计一个控制器，使得系统$\dot{x} = f(x)$能够从任意的状态x_0出发，当$t \to \infty$时，系统的状态$x(t) \to x(t)_d$。

随时间变化的轨迹

非完整机器人只有两个输入$[v \quad \omega]$，而其构型空间$[\phi \quad x \quad y]$却是三维的，因此不可能像全向移动机器人那样独立的控制构型空间的每一维。因此，对于非完整移动机器人，我们说的轨迹跟踪实际上是控制机器人上任意的一点P，**跟随二维平面$[x \quad y]$上的一条轨迹**。





将局部坐标转换为全局坐标

$$\begin{bmatrix} x_p \\ y_p \end{bmatrix} = \begin{bmatrix} x \\ y \end{bmatrix} + \begin{bmatrix} cos\phi & -sin\phi \\ sin\phi & cos\phi \end{bmatrix} \begin{bmatrix} x_r \\ y_r \end{bmatrix}$$

求一次导:

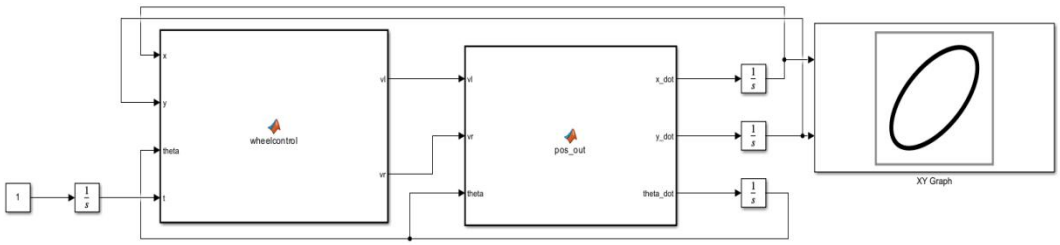
$$\begin{bmatrix} \dot{x}_p \\ \dot{y}_p \end{bmatrix} = \begin{bmatrix} \dot{x} \\ \dot{y} \end{bmatrix} + \dot{\phi} \begin{bmatrix} -sin\phi & -cos\phi \\ cos\phi & -sin\phi \end{bmatrix} \begin{bmatrix} x_r \\ y_r \end{bmatrix}$$

最简单的**P**控制器:

$$\begin{bmatrix} \dot{x}_p \\ \dot{y}_p \end{bmatrix} = K \begin{bmatrix} x(t)_d - x_p \\ y(t)_d - y_p \end{bmatrix}$$

$$\begin{bmatrix} v \\ \omega \end{bmatrix} = \frac{K}{x_r} \begin{bmatrix} x_r cos\phi - y_r sin\phi & x_r sin\phi + y_r cos\phi \\ -sin\phi & cos\phi \end{bmatrix} \begin{bmatrix} x(t)_d - x_p \\ y(t)_d - y_p \end{bmatrix}$$

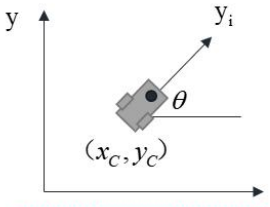
四、控制器设计（轨迹跟随与镇定控制）



因为Brockett的结论，人们放弃了寻找连续反馈控制律的尝试，转而寻找不连续或者时变的控制。

2.1 镇定控制

J.-B. Pomet等人提出了一种控制器：



$x_1 = x_c \cos\theta + y_c \sin\theta$ ，径向位移

$x_2 = \theta$ ，

$x_3 = x_c \sin\theta - y_c \cos\theta$ ，切向位移

$u_1 = v - wx_3$ ，径向速度

$u_2 = w$.

$u_1(x, t) = -x_1 + x_3(\sin t - \cos t)$ → 时变光滑

$u_2(x, t) = -x_2 = x_1x_3 - x_1(x_1 + x_3 \cos t) \cos t$

2.1 镇定控制

非完整系统难以镇定的原因在于系统受到了约束。而非完整系统具有**微分平坦**的性质。可以将系统从状态空间转换到平坦输出空间，构造一个多阶积分器系统，从而方便的控制系统。

存在这样一个系统(线性或非线性):

$$\dot{\boldsymbol{x}} = f(\boldsymbol{x}(t), u(t))$$

- 系统存在一个输出$F = h(\boldsymbol{x})$使得其状态$\boldsymbol{x}(t)$和输入$u(t)$可以表示为:

$$\boldsymbol{x}(t) = \boldsymbol{\phi}_{\boldsymbol{x}}(F, \dot{F}, \ddot{F}, \ldots, F^{(n-1)})$$
$$u = \phi_u(F, \dot{F}, \ddot{F}, \ldots, F^{(n-1)}, F^{(n)})$$

其中$\boldsymbol{\phi}_{\boldsymbol{x}}, \phi_u$都为光滑函数

则称其为微分平坦系统，$F(t)$则称为系统的平坦输出。

控制器设计：

轨迹跟踪的公式是：

$$\begin{aligned} v &= v_{max}\cos\alpha \\ w &= \frac{v_{max}\sin\alpha}{d} \\ v_l &= v - dw \\ v_r &= v + dw \end{aligned}$$

其中，v 是车辆的线速度，w 是车辆的角速度，v_l 和 v_r 是左右车轮的速度，v_{max} 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 (x_0, y_0) 和终点 (x_1,y_1) 的坐标

终点车辆的朝向 θ_{goal}

车轮参数 d 和最大速度 v_{max}

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

2. 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：

$$q = [x, y, \theta]^T$$

（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：

$$\dot{x}\sin\theta - \dot{y}\cos\theta = 0$$

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

$$\dot{x} = v\cos\theta \quad \dot{y} = v\sin\theta \quad \dot{\theta} = \omega$$

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

$$\dot{x}_d = v_d\cos\theta_d \quad \dot{y}_d = v_d\sin\theta_d \quad \dot{\theta}_d = \omega_d$$

$$q_d = [x_d, y_d, \theta_d]^T$$

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：

$$\lim_{t\to\infty}(q(t) - q_d(t)) = 0$$

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 X_r、Y_r

在 x,y,θ方向上的误差分别为：

$$e_x = x_r - x, e_y = y_r - y, e_\theta = \theta_r - \theta$$

定义位姿跟踪误差：

$$q_e = \begin{bmatrix} x_e \\ y_e \\ \theta_e \end{bmatrix} = \begin{bmatrix} \cos\theta & \sin\theta & 0 \\ -\sin\theta & \cos\theta & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} x_r - x \\ y_r - y \\ \theta_r - \theta \end{bmatrix}$$

求导得到：

$$\begin{aligned} \dot{x}_e &= -\dot{\theta}\sin\theta\,(x_r - x) + \cos\theta\,(\dot{x}_r - \dot{x}) + \dot{\theta}\cos\theta\,(y_r - y) + \sin\theta\,(\dot{y}_r - \dot{y}) \\ &= \dot{\theta}\left[-\sin\theta\,(x_r - x) + \cos\theta\,(y_r - y)\right] - v + v_r\,(\cos\theta_r\cos\theta + \sin\theta_r\sin\theta) \\ &= \omega y_e - v + v_r\cos\theta_e \end{aligned}$$

$$\begin{aligned} \dot{y}_e &= -\dot{\theta}\cos\theta\,(x_r - x) - \sin\theta\,(\dot{x}_r - \dot{x}) - \dot{\theta}\sin\theta\,(y_r - y) + \cos\theta\,(\dot{y}_r - \dot{y}) \\ &= -\dot{\theta}\left[\cos\theta\,(x_r - x) + \sin\theta\,(y_r - y)\right] + v_r\,(-\cos\theta_r\sin\theta + \sin\theta_r\cos\theta) \\ &= -\omega x_e + v_r\sin\theta_e \end{aligned}$$

整理位姿误差微分方程为：

$$\begin{bmatrix} \dot{x}_e \\ \dot{y}_e \\ \dot{\theta}_e \end{bmatrix} = \begin{bmatrix} \omega y_e - v + v_r\cos\theta_e \\ -\omega x_e + v_r\sin\theta_e \\ \omega_r - \omega \end{bmatrix}$$

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 $q = (vw)^T$ 使闭环系统位姿误差P_e 凡能够全局一致有界，并且当 $t \to \infty$，系统在任意的初始跟踪误差下有

$$\lim_{t\to\infty}\left[|x_e(t)| + |y_e(t)| + |\theta_e(t)|\right] = 0.$$

根据自主导航车的位姿误差微分方程，可以选取函数如下：

$$V = \frac{k_1}{2}\left(x_e^2 + y_e^2\right) + 2\left(\sin\frac{\theta_e}{2}\right)^2 \geq 0$$

将其微分得：

$$\begin{aligned} \dot{V} &= k_1\left(x_e\dot{x}_e + y_e\dot{y}_e\right) + 2\sin\frac{\theta_e}{2}\cos\frac{\theta_e}{e}\dot{\theta}_e \\ &= k_1\left[x_e\left(v_r\cos\theta_e - v + y_e\omega\right) + y_e\left(v_r\sin\theta_e - x_e\omega\right)\right] + (\omega_r - \omega)\sin\theta_e \\ &= k_1\left[x_e v_r\cos\theta_e - v x_e\right] + k_1 y_c v_r\sin\theta_e + (\omega_r - \omega)\sin\theta_e \end{aligned}$$

由此可得控制律为：

$$q = \begin{bmatrix} v \\ w \end{bmatrix} = \begin{bmatrix} v_r\cos\theta_e + k_2 x_e \\ w_r + k_1 v_r y_e + k_3\sin\theta_e \end{bmatrix}$$

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。

```
while(flag ~=1)
     [i_min,flag] = min_fn(OPEN,OPEN_COUNT,xTarget,yTarget);
    node_x = OPEN(i_min,1);
    node_y = OPEN(i_min,2);

    gn = OPEN(i_min, 4);

    [exp_array,~] =expand_array(node_x,node_y,gn,xTarget,yTarget,CLOSED,OPEN,MAX_X,MAX_Y);
```

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。

```
for ce=1:size(exp_array,1)
     %if ce in open, continue. or add it to open
     in_open=0;
    for c1=1:OPEN_COUNT

        if(exp_array(ce,1) == OPEN(c1,1) && exp_array(ce,2) == OPEN(c1,2))
            in_open=1;
            if exp_array(ce,5) < OPEN(c1,5)
                OPEN(c1,:) = exp_array(ce,:);
            end
       break
        end

    end%End of for loop to check if a successor is on closed list.
     if (in_open==0)
         OPEN_COUNT=OPEN_COUNT+1;
          OPEN(OPEN_COUNT,:)=exp_array(ce,:);
     end
end
```

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。

```
traj_path_search(i,:) = [OPEN(i_min,1:2) OPEN(i_min,6:7)];

OPEN(i_min,:) = [ ];
CLOSED_COUNT=CLOSED_COUNT+1;
CLOSED(CLOSED_COUNT,1)=node_x;
CLOSED(CLOSED_COUNT,2)=node_y;

OPEN_COUNT = size(OPEN,1);
```

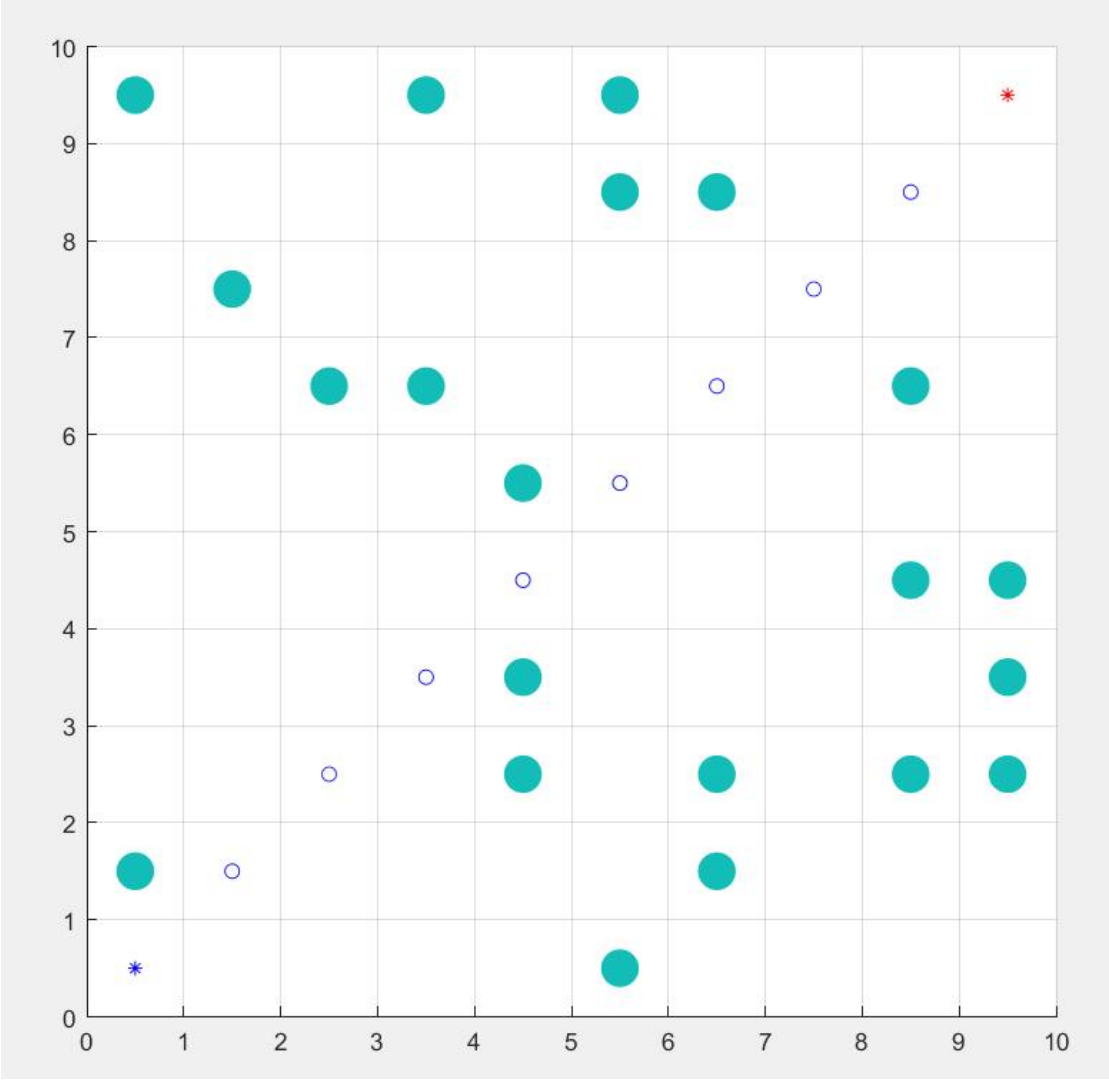
拓展结束后，将对应被拓展的节点加入到关闭列表中。

```
aj = traj_path_search ;
th(1,:) = traj (end,1:2);
=1;
in = traj(end,:);
ondi = 0;
hile condi ==0
r i = 1:length(traj)

  if traj(i,1) == pin(3) && traj(i,2) ==pin(4
      path(k+1,:) = [pin(3), pin(4)];
       k = k+1;
       pin = traj(i,:);
  end
d
if pin(1)== traj(1,1) &&pin(2) ==traj(1,2)
      condi =1;
end
nd
th = [[xTarget, yTarget];path(1:end,:)]
```

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；

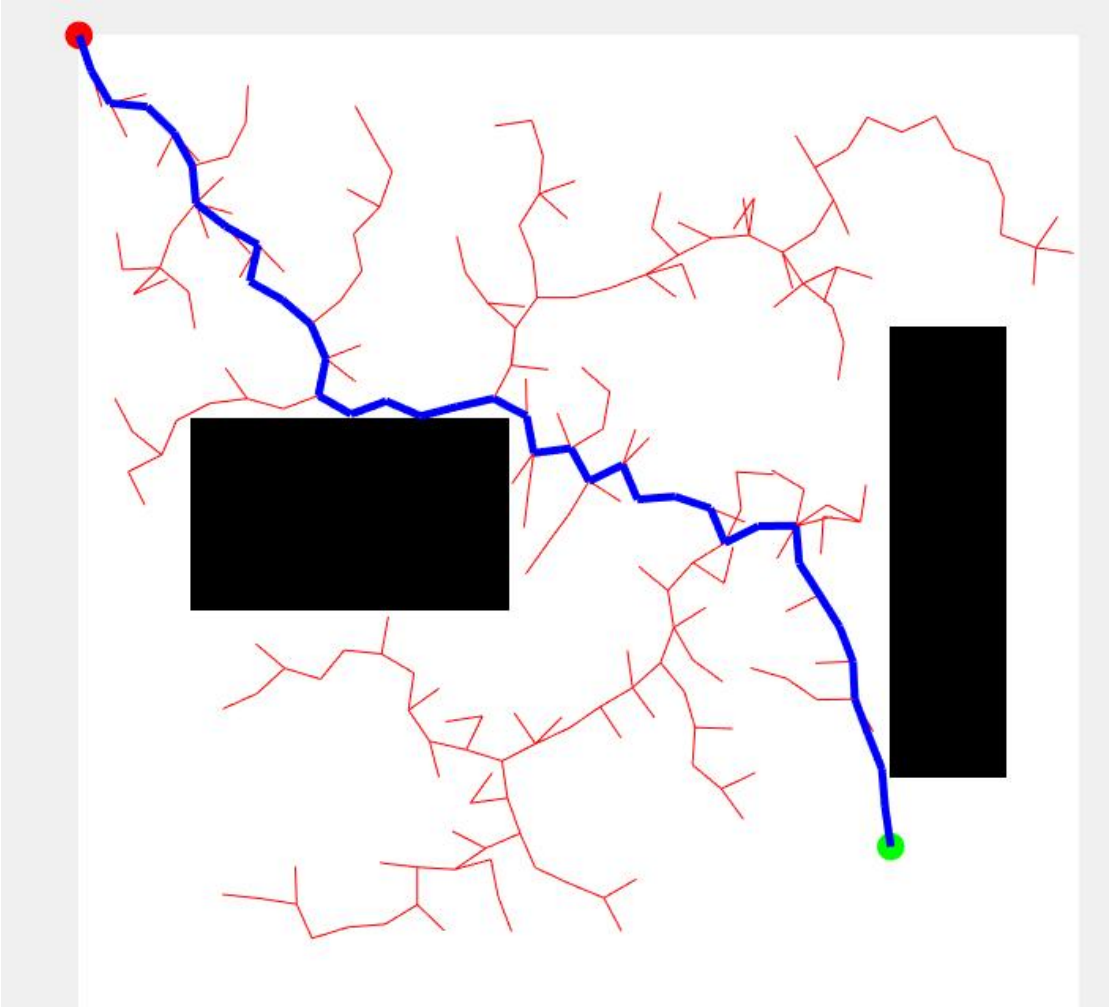
（2）可能出现计算复杂、所需的时间过长、易于陷入死区的问题；

（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。

5、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。

```
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);
```

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。

```
%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);
```

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。

```
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
```

```
88  -       while abs(x-x_flag - x_goal) > dis_err || abs(y-y_flag - y_goal) > dis_err
89  -           x=x_flag
90  -           y=y_flag
91  -           theta=theta_flag
92  -           dx=x_goal-x+x_flag;
93  -           dy=y_goal-y+y_flag;
94  -           alpha=atan2(dy,dx+err)-theta;
95  -           alpha=atan2(sin(alpha),cos(alpha)+err);
96              % vel.Linear.X=0.1;
97              % vel.Angular.Z=0.1;
98  -           vel.Linear.X = v_max*cos(alpha);
99  -           vel.Angular.Z = v_max*sin(alpha)/d;
100             % vel.Linear.X=0.6;
101             % vel.Angular.Z=0;
102 -           send(vel_pub, vel);
103 -           msg2 = receive(odom_sub, 10); % 接收里程计消息，最长等待时间为 10 秒
104 -           x = msg2.Pose.Pose.Position.X;
105 -           y = msg2.Pose.Pose.Position.Y;
106 -           quat = msg2.Pose.Pose.Orientation;
107 -           angles = quat2eul([quat.W quat.X quat.Y quat.Z]);
108 -           theta = angles(1);
109
110 -       end
111 -       i=i-1;
112 - end
```

6、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

A. 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,yTarg
et,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
```

B. RRT 算法

```
# collisionChecking.m

function
feasible=collisionChecking(startPose,goalPose,map)


feasible=true;
dir=atan2(goalPose(1)-startPose(1),goalPose(2)-startP
ose(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
```

C.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/veloci
ty', '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
```

D. 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
```