队伍编号	MC2204132
题号	C

基于 0 - 1 规划模型的自动泊车问题研究

摘　要

自动泊车是自动驾驶技术中使用需求最多的场景之一，在停车空间有限的大城市，是一个比较实用的功能，减少了驾驶员将车辆驶入狭小空间的难度。

对于问题一：以无人乘用车为例，我们根据题目中的方向盘最大转角及车辆长度，得出车辆最小转弯半径$4.99m$；接着根据限制车辆最大加加速度，得出从最短加速到最大限制速度的距离$1.38m$；根据最短距离、最小转弯半径及方向盘与前轮转角的传动比，最终得出路径上的曲率相对路径长度的变化率小于$0.212s^{-1}$。

对于问题二：根据不得与旁边车位发生碰撞的极值点，分析可得其最短转弯方式，并以此求得其初始位置离车库顶部垂直距离为$1.13m$。再由于其宽度距离大于$5.30m$，由转弯半径分析可得，无人车刚好到库进行倒车。解极坐标方程组，判断出其先做减速运动，再做加速圆周运动，最后再做减速直线运动。对于倾斜车位，同理可得极坐标方程组。对于平行入库，其先向右做匀速圆周运动，再向左做圆周运动。利用中间转折点性质，可求得其左右两侧圆周运动得极坐标方程。运用 Matlab 得出泊车轨迹并模拟泊车过程。

对于问题三：我们以停车场入口为原点建立坐标系，以无人车位置坐标为未知量，通过四个变量：停车位的 x、y 位置、无人车在该位置倒车前的方向以及倒车过程所需旋转角度来设置停车位的坐标。由于无人车运动过程较为复杂，我们考虑直线路径行驶，始终为最大速度，加速带前后五米距离始终取极限速度，最后通过构建目标函数和约束条件解得在图 3 无人车位置的情况下，13 号停车位为最优解。也凭借此坐标系，可用 Python 求得无人车到各个停车位所需时间，并用其筛选得出最短时间即为最优解。

对于问题四：基于问题三的基础上，我们用 python 设计程序，实现开始随机 30 个车位被占用，再假设车辆一进一出时间相同，实现一进一出车位随机，给占用车位赋值 0，再通过问题三的程序算出到空余车位所需的时间，筛选出需要最短时间的车位，该车位即为最优解。

关键词：最小转弯半径 车辆运动极坐标方程 泊车模型 0-1 规划模型

目录

一、 问题重述 3
1.1 问题背景 3
1.2 目标任务 3
二、 模型假设及符号说明 4
2.1 模型假设 4
2.2 主要符号说明 4
三、 问题分析 5
3.1 针对问题一 5
3.2 针对问题二 5
3.3 针对问题三 5
3.4 针对问题四 6
四、 问题一模型的建立与求解 6
4.1 车辆最小转弯半径的计算 6
4.2 加速到最短限制速度的最短距离 6
4.3 曲率相对路径长度的变化率的限制 7
五、 问题二模型的建立与求解 8
5.1 10 号泊车位泊车模型 8
5.1.1 泊车模型极点情况 8
5.1.2 垂直泊车模型通解 9
5.2 82 号车位泊车模型 10
5.3 31 号车位泊车模型 13
六、 问题三模型的建立与求解 14
6.1 选择最优泊车位的 0-1 规划模型 14
6.2 模型求解 16
七、 问题四模型的建立与求解 16
7.1 随机进出泊车模型 16
7.2 模型的求解 17
八、 模型的评价与改进 17
8.1 模型的优点 17
8.2 模型的缺点 17
8.3 模型的改进 17
参考文献 18
附录 18

一、 问题重述

1.1 问题背景

随着科技的发展，自动驾驶是近年来人工智能应用的热门研究领域，其中自动泊车是自动驾驶技术中落地最多的场景之一，自动泊车指的是在停车场内实现无人车的自动泊车入位过程，对于停车空间有限的大城市，这是一个比较实用的功能，既减少了驾驶员将车辆驶入狭小空间的难度，又可以节约空间的使用，实现最大利益化。自动泊车在不需要人工控制的情况下，主要通过遍布车辆周围的传感器测量自身与停车位之间的距离和角度，还包括与该场景中其他障碍物之间的相对位置，然后通过车载电脑计算出操作流程，配合着车速调整方向盘的转动，让无人车自动泊车入位，对所有驾驶员来说，这是一项相当便捷的配置。因此，自动泊车的研究成为了车辆自动驾驶的研究重点之一。

1.2 目标任务

问题一：根据给出的无人车模型的参数，计算车辆最小转弯半径；限制车辆最大加加速度为 20 m/s，无人车沿直线行驶时，加速到最大限制速度的最短距离；当车速为 20km/h 时，无人车需要从沿直线行驶状态转弯，路径上的曲率相对路径长度的变化率的限制。

问题二：建立无人车从车库入口泊车的数学模型，给出从车库入口到指定停车位的泊车轨迹，轨迹包括每时刻无人车的行驶路径长度、车辆朝向、速度、加速度、加加速度、角速度、角加速度等，并做出可视化轨迹图。在泊车过程中标注红色禁停的停车位都已经被占用，泊车过程中无人车不能与其发生冲突或碰撞。分别考虑三种不同的车位情况，10 号垂直停车位、8 号平行泊车位、31 号倾斜停车位（倾斜角为 45°）。

问题三：根据当前停车位的状况，建立无人车在初始位置上的泊车模型，计算出最优停车位，画出当前位置到停车位的轨迹；标注红色禁停的停车位都已经被占用，泊车过程中无人车不能与其发生冲突或碰撞。建立这个过程的通用模型，并考虑算法，设计出能适应车库中任意停车位被占用的状况，考虑这

个过程算法复杂性。

问题四：在当前状态下假设每小时内从入口进入和从出口离开停车场的车辆均为 30 辆，因车辆的进入和离开，导致停车位会被随机占用或释放。为无人车建立泊车模型，并给出从当前位置到最优停车位的行驶轨迹的仿真结果。

二、 模型假设及符号说明

2.1 模型假设

为了便于考虑问题，我们在不影响模型准确性的前提下，作出以下假设：

（1）假设加速阶段为匀加速运动，减速阶段为匀减速运动；

（2）假设第三问无人车泊车速度均为最大速度。

（3）假设所有转弯过程运动轨迹均为圆弧。

（4）由轮胎与地面的摩擦消耗的时间忽略不计。

（5）假设第四问车辆一进一出间隔时间相同。

2.2 主要符号说明

注：此为本文的主要符号说明，其它符号解释详见正文部分。

符号	意义
α_1	前轮最大转角（°）
L	车身长度（m）
d	车身宽度（m）
k	车辆的加加速度（m/s^3）
$Xmax$	最短距离（m）
$Vmax$	最大限制速度（m / s）
φ	曲率（m^{-1}）
D_1	停车场宽度（m）

三、 问题分析

3.1 针对问题一

本题目的在于根据题目以及附件所给信息，计算车辆最小转弯半径、加速到最大限制速度的最短距离、曲率相对路径长度的变化率的限制。首先，我们根据题目中的车长、车辆方向最大转角，计算出车辆最小转弯半径 ；然后再通过题目中的最大限制速度，计算出加速到最大限制速度的最短距离；最后通过，计算出车速为最大限制速度时，无人车从沿直线行驶状态开始转弯，路径上的曲率相对路径长度的变化率的限制，这个过程中运用 Maple 软件仿真无人车泊车绘图。

3.2 针对问题二

本题对于 10 号车位，我们考虑它不能与右边的车位发生碰撞，所以此时能取到一个特殊极点。根据这个极点，我们可以得出车的初始位置以及运动轨迹。我们考虑倒车入库有两种情况：一种是转弯后刚好抵达车库位置，第二种是在转弯过程中到达车库最底部。由于宽度距离大于 5.3，和所求转弯半径分析得，车刚好到库进行倒车，由此可推断出它在 10 号车位的运动方式。针对 82 号车位，我们考虑路径中值左右两路径关于中值点中心对称。由此确立无人车运动轨迹极坐标方程。针对 31 号倾斜停车位，采取方法与 10 号垂直停车位类似，即建立坐标轴倾斜 45°求解。最后，我们采用 Matlab，通过极坐标方程画出对应轨迹，再用 AE 软件绘图显示更明确的结果。

3.3 针对问题三

问题三是要求我们建立通用泊车最优模型，我们以入口为原点建立坐标系本题我们以停车场入口建立坐标系，以无人车位置坐标为未知量，通过四个变量：停车位的 x，y 位置以及无人车在该位置倒车前的方向，和倒车过程所需旋转角度设置停车位的坐标，进而我们列出分三段无人车运动的时间，一段为倒

车时间，一段为弯道时间，一段为直线距离，最后通过构建目标函数时间总和和约束条件泊车位是否空余解得在图 3 无人车位置的情况下，xx 停车位为最优解。也凭借此坐标系，可用 Python 建立通用模型，即可得到不同无人车位置到不同泊车段所需要的时间，最后再用 python 筛选出最少时间的泊车位即为最优解。其中停车位坐标列在 excel 中，附件在计算结果。

3.4 针对问题四

求解方式与问题三类似，区别在于它需要随机生成停车位，并编写程序作出在相同时间间隔内车辆一进一出所占用和离开的车位，对其进行 0-1 规划，再结合问题三求出到达空余车位所需要的时间，筛选出最短时间，则最短时间对应的车位就为最优泊车位，泊车轨迹与第二题三种情况对应相同。

四、 问题一模型的建立与求解

4.1 车辆最小转弯半径的计算

首先，根据题目所给条件，方向盘最大转角为 470°，方向盘与前轮转角的传动比为 16：1，也就是方向盘转动 16°，前轮转动 1°，因此可求得前轮最大转角为α1=470°/16=29.375。由于车长 L=4.9m。由调查研究可知无人车的最小转弯半径[4]为：

$$Rmin=L/2\sin(\alpha 1)$$

所以带入数据得到

$$Rmin=4.9947m$$

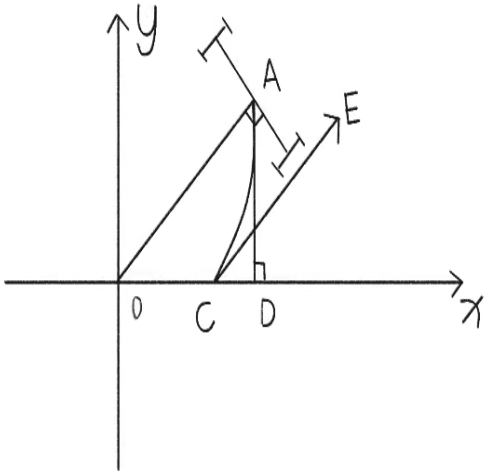
4.2 加速到最短限制速度的最短距离

由题目得知，最大加加速度 aa 为 20m/s³，最大限制速度 Vmax 为 20km/h，转化单位为 Vmax=5.556m/s。。由查阅资料得知，最短距离是在最大加加速度时取到，即对加加速度取三次积分，即可求得最短距离。

aa=da/dt=20，即 da=20*dt，两边同时求积分可得 a=20*t。由于 a=dVmax/d

t，两边同时求积分可得 $Vmax=10*t^2$。由于 Vmax=dx/dt，两边同时求积分可得 $x=10/3*t^3$。由 $Vmax=10*t^2$，可得到 t= √5/3s，带入 $x=10/3*t^3$ 可得最短距离 x= 50 √5/81m。

4.3 曲率相对路径长度的变化率的限制



曲率就是针对曲线上某个点的切线方向角对弧长的转动率，通过微分来定义，表明曲线偏离直线的程度。数学上表明曲线在某一点的弯曲程度的数值。曲率越大表示曲线弯曲的程度越大。

设 AC 的弧长为 S，S∈[0,x]。

设∠AOC=θ，查阅资料可知θ=S/Rmin。

设 P 为最大曲率，P=|（∠ADO-∠ACE）/S|=|(π/2-θ-α1)/S|即φ<P。

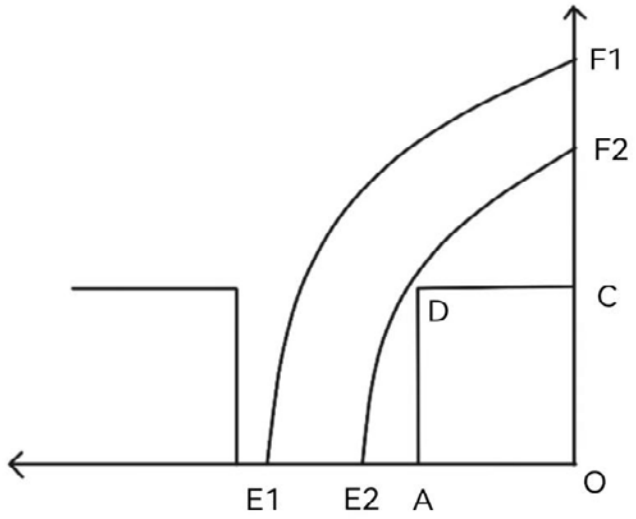
五、 问题二模型的建立与求解

5.1 10 号泊车位泊车模型

5.1.1 泊车模型极点情况

我们考虑无人车泊车恰好经过 D 点，该点即为另一车位的顶点，此为极值情况，列出单独计算。

由图可得，



$$BF_1 = BE_1 = R$$

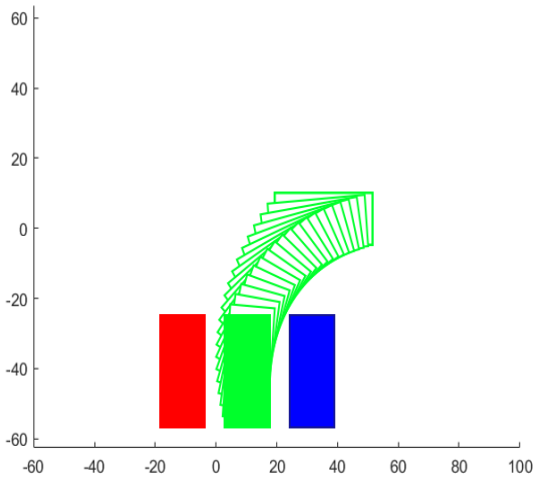
$$E_2A = X$$

$$x = R - d - OA$$

$$OA = \sqrt{2}/2(R - d)$$

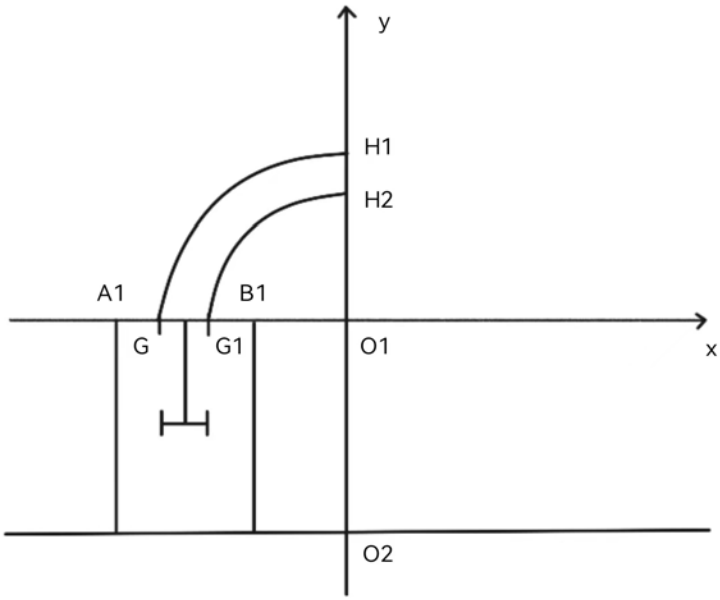
$$CF_2 = x = \sqrt{2}/2(R - d) = 1.1295$$

由此可确定，车的初始位置为 CF_2,并可得到行车轨迹，Matlab 作图得：



5.1.2 垂直泊车模型通解

由于垂直倒车存在两种情况，一种是转弯后刚好抵达车库位置，第二种是在转弯过程中到达车库最底部。由于宽度距离大于 5.3，和所求转弯半径分析[3]得，车刚好到库进行倒车，则作图得

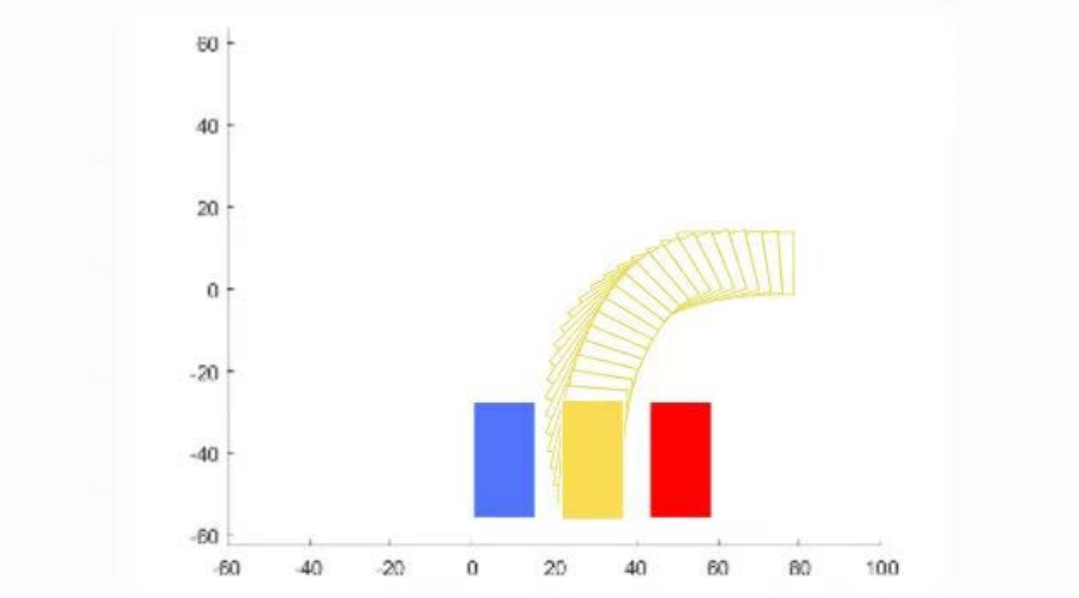


若 OE 坐标为（0，0），则得到弦 GH1 的极坐标方程为

$$\begin{cases} x = -Rsin\theta \\ y = Rcos\theta \end{cases}$$

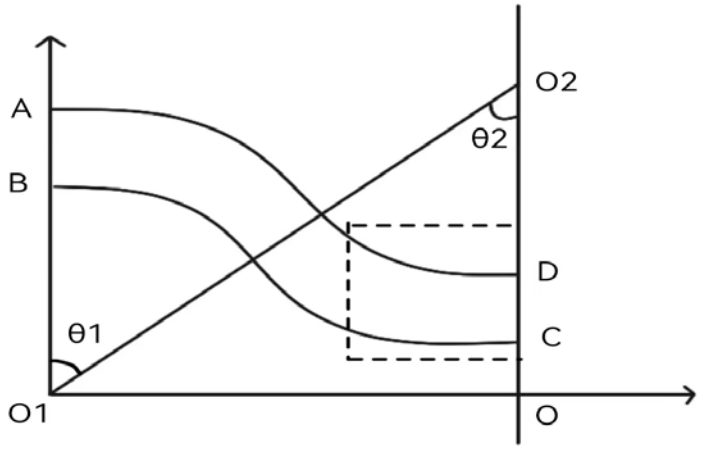
当以 02 作为原点，B1O1=R-d -（A1B1-d）/2，由此可推出 H1H2 的坐标。其运动方式为，先加速到最短距离 50 √5 \\8\1 沿直线路径方向，再接最大速度 20km/h 匀速圆周运动，朝向为轨迹切线，加加速度为 20km/h，角速度为 v/Rmin，角加速度为 0，刚好入库时减速，a_{max}=6m/s2，t=v/a_{max}，x=vt-1/2a_{max}*t2，

方向为进 10 号车库方向，用 Matlab 绘制极坐标运动轨迹方程得出如图泊车过程。



5.2 82 号车位泊车模型

由车运动图



可知 θ1=θ2

可知 O_1 O_2=2R-d

82 号停车场变为 d1

O2O=R+（d1-d/2）

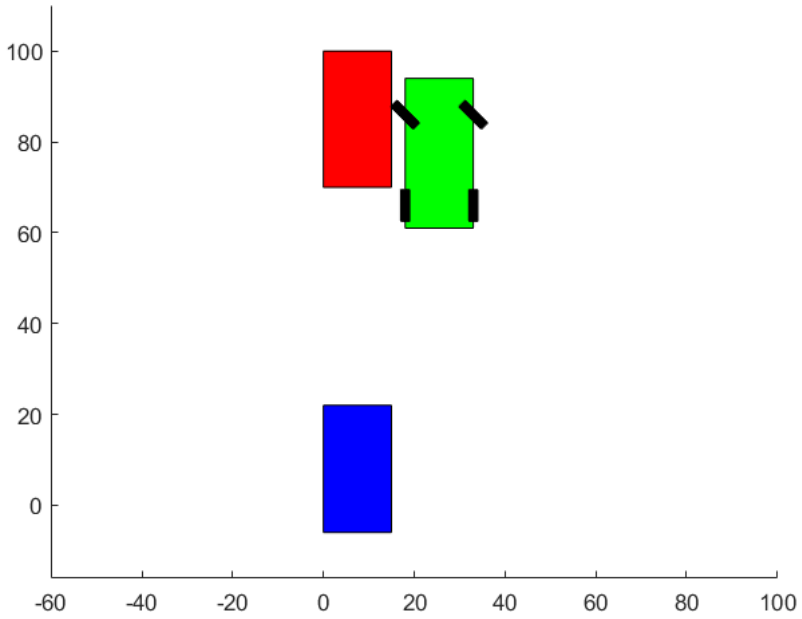
可知曲线左边段 O1A

$$\begin{cases} x = -\sqrt{(2R-d)^2 - \left(R + \dfrac{(d_1 - d)}{2}\right)^2} + \sin\theta \\ y = \cos\theta\, R \end{cases}$$

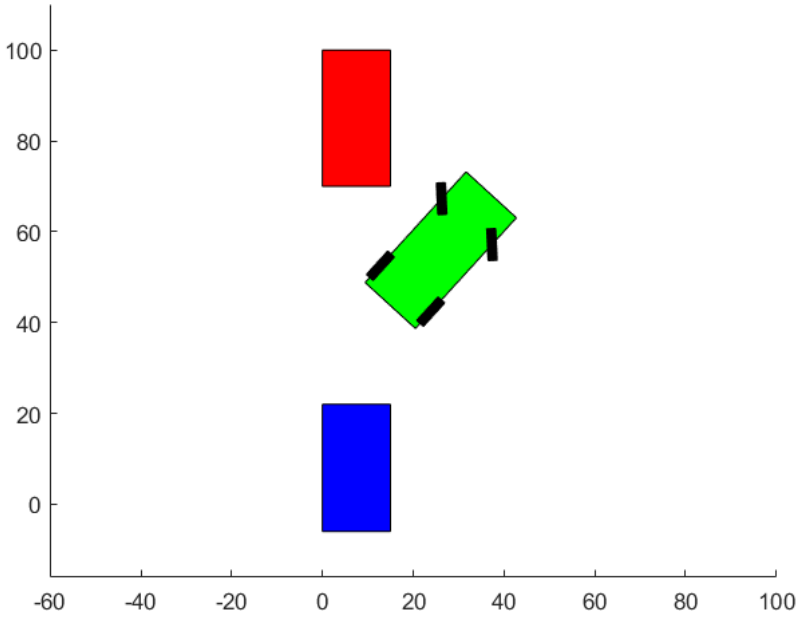
在 AO 段曲线右侧部分

$$\begin{cases} x = -\sqrt{R - d - \sin\theta(R-d)} \\ y = \cos\theta\, R \end{cases}$$

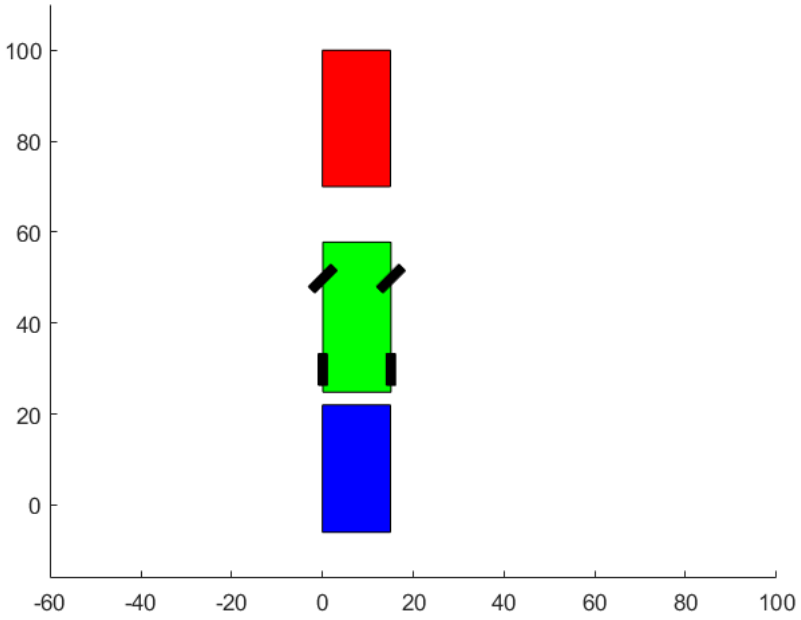
由两个极坐标轨迹公式，通过 Matlab 绘图并分析出轨迹图如下：



步骤 1


100
80
60
40
20
0
-60
-40
-20
0
20
40
60
80
100

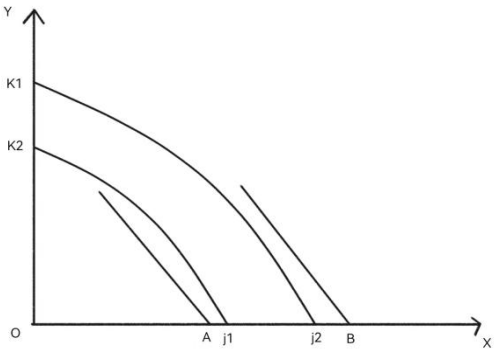
步骤 2


100
80
60
40
20
0
-60
-40
-20
0
20
40
60
80
100

步骤 3

5.3 31 号车位泊车模型

分析如 10 号平行泊车模型一致，倒车轨迹运动草图如下：



由运动草图可知

oj2=R

Aj1=(d1-d)/2

oA=R-d-(d1-d)/2

因为 45°斜角

所以 oK=oA=R-d-(d1-d)/2

由此确立估计极坐标方程

x=oK*sinθ

oK2 弧为

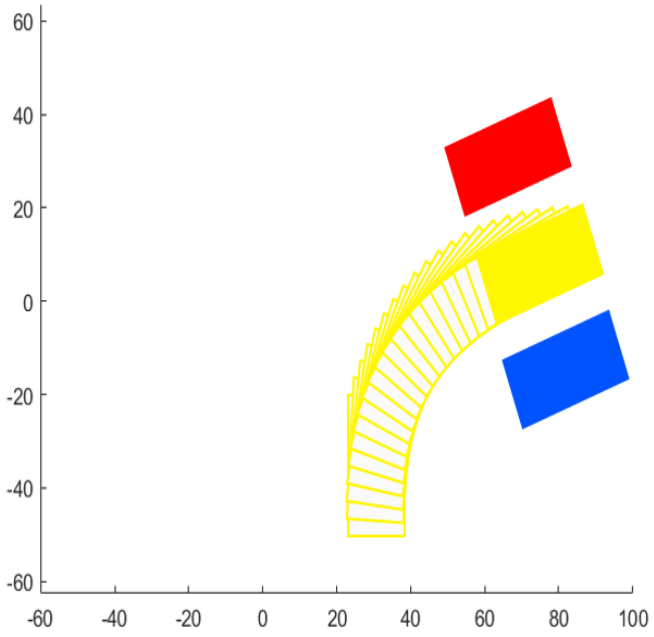
y=oK*cosθ

x=R*sinθ

oK1 弧为

y=R*cosθ

由此极坐标方程可用 Matlab 求解轨迹，作轨迹图如下：



六、 问题三模型的建立与求解

6.1 选择最优泊车位的 0-1 规划模型

假设所有路段 V 都为最大值

车位坐标为$(x_i，\ y_i，\ \alpha_i，\ \beta_i)$

α_i:无人车倒车前一段方向

β_i：无人车倒车过程需旋转方向

设弯道改为k_i

$$k_i=\begin{cases}0，\ \alpha_i=0\\ 1，\ \alpha_i=\dfrac{\pi}{2}\\ 2，\ \alpha_i=\pi\end{cases}$$

弯道路径$S_{1i}=2\pi R_{min}\cdot k_i，\ V_1=V_{max}=20km/h$

$$t_i=\frac{S_{1i}}{V_{max}}$$

直线$\triangle x_i=x-x_i,\triangle y_i=y-y_i$

$$S_{2i}=\begin{cases}|\triangle x_i|，\ \alpha_i=0\\ \triangle x_i+\triangle y_i,\alpha_i=\dfrac{\pi}{2}\\ 2S_{max}-(x+x_i)，\ \alpha_i=\pi\end{cases}$$

设在减速带 5m 时，$V_2=\frac{10km}{h}$，*其他时段*$V_3=V_{max}=20km/h$

设最远距离为$S_{max}=70.1m$

当$\alpha_i=0$ *时*

$$t_{2i}=\begin{cases}\dfrac{S_{max}-x-5}{V_{max}}+\dfrac{S_{max}-x+\triangle y_i}{V_2},S_{max}-x>5，\ \triangle x_i>0\\ \dfrac{S_{max}-x+\triangle y_i}{V_2},S_{max}-x\le 5，\ \triangle x_i>0\\ \dfrac{-\triangle x_i}{V_2},\triangle x_i<0\end{cases}$$

当$\alpha_i=\pi/2$ *时*

$$t_{2i}=\begin{cases}\dfrac{S_{max}-x-5}{V_{max}}+\dfrac{S_{max}-x+\triangle y_i}{V_2},S_{max}-x>5，\ \triangle x_i>0\\ \dfrac{S_{max}-x+\triangle y_i}{V_2},S_{max}-x\le 5，\ \triangle x_i>0\end{cases}$$

当$\alpha_i=\pi$*时*

t_{2i}

$$=\begin{cases}\dfrac{S_{max}-x-5}{V_{max}}+\dfrac{S_{max}-x+\triangle y_i}{V_2}+\dfrac{5}{V_2}+\dfrac{S_{max}-x_i-5}{V_{max}},S_{max}-x_i\ge 5,S_{max}-x>5\\ \dfrac{S_{max}-x+5}{V_{max}}+\dfrac{S_{max}-x+\triangle y_i}{V_2}+\dfrac{b-x_i}{V_2},S_{max}-x_i\le 5,S_{max}-x>5\\ \dfrac{S_{max}-x+\triangle y_i}{V_2}+\dfrac{5}{V_2}+\dfrac{S_{max}-x_i-5}{V_{max}},S_{max}-x_i>5,S_{max}-x\le 5\\ \dfrac{S_{max}+x+\triangle y_i}{V_2}+\dfrac{5}{V_2}+\dfrac{S_{max}-x_i}{V_2},S_{max}-x_i\le 5,S_{max}-x\le 5\end{cases}$$

倒车路径$S_{3i}=2\pi R\cdot\beta_i$

$$t_{3i}=\frac{S_{3i}}{V_2}$$

设对 n 个停车位决策系改为a_i

目标函数$t_i = a_i（t_{1i} + t_{2i} + t_{3i}）$

约束条件$\sum a_i = 1$

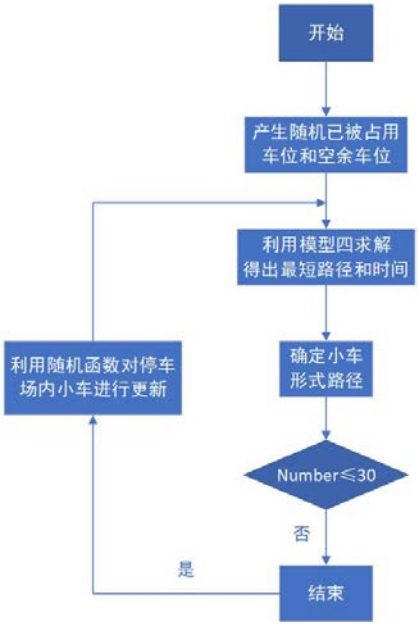
6.2 模型求解

我们通过建立坐标系，建立各变量函数关系，通过 python 代码编写得到无人车位置到各个泊车位所需的时间[2]，针对该题，我们可以很直观地得出 13,45,67,78 四个车位有可能时间最短，即为最优停车策略，因此我们用 python 对这四个车位求解。首先 13 号车位坐标为（31.2,0,0,0），所需时间为 6.048 秒，再是 45 号车位坐标为（38.4,16.2， π，1/2 π），所需时间为 38.77 秒，67 号车位坐标为（30.6，5.5,0,1/2 π），所需时间为 7.568 秒，最后 78 号车位坐标为（35.7,10.7, π ,1/2 π），所需时间为 41.24 秒，综上，13 号车位为最优停车位，其泊车轨迹即同第二题垂直泊车位轨迹相同。

七、 问题四模型的建立与求解

7.1 随机进出泊车模型

我们假设车的进出间隔时间相同为△t，停车场变化过程如下图所示：



因此，进出车辆互不影响且进出车辆泊车位随机生成，凭此，我们用 python 建立随机生成模型。

7.2 模型的求解

我们通过 python 代码，随机生成车辆位置占有，再确定三十辆车的最优泊车位，再形成泊车轨迹[1]，得出最后的最优解。

八、 模型的评价与改进

8.1 模型的优点

1.该模型采用分段函数，可分段计算出行车路径，准确率较高，可以更高效率的解得目标函数。

2.该模型适用范围广，不仅局限于题目所给的限制，适用于多辆车多种情况。

3.采用 matlab 绘制无人车路径图像，路线平滑，较为准确。

4.建立模型物理学知识偏多，所得数据更为准确，误差较小.

8.2 模型的缺点

1.计算路径忽略了无人车转弯速度变化，存在一定的误差

2.未考虑实际环境的影响，无人车内因外因都有可能影响最后实际的运行。

3.在该模型中，路径过程没有考虑变化的物体量，忽略了动态因素。

8.3 模型的改进

1.针对外界环境影响，我们可分类讨论，如天气因素和其他车辆存在的因素等，分别对模型产生的影响，优化模型，尽可能减小外界因素的干扰。

2.结合外界交规，对无人车速度进行限速，更符合安全规定。

3.考虑采用插值拟合算法为无人车进行路径规划。

参考文献

[1]王浩杰　大连理工大学　自动泊车系统路径规划与跟踪算法研究　2021-12-16

[2]罗奕树 张在龙 王日信 闽江学院学报. 自动泊车小车转弯参数及特征环境参数的确定　2011,32(05)

[3]Robust automatic parallel parking in tight spaces via fuzzy logic[J] . Yanan Zhao,Emmanuel G. Collins. Robotics and Autonomous Systems . 2005 (2)

[4]张道权 程翔 柯为 唐皓冲 广州汽车集团股份有限公司汽车工程研究院 减小最小转弯半径优化及验证 2022.004.022

附录

所用软件：Python、Matlab

问题二：

```
clc
close all
clearvars
//全局变量
global l;
global d;
global Dc;
global Lc;
global W;
l=4.9;
d=1.8;
   A=15;
   Lc=20;
   Dc=5;
   F=8;
   B=Lc+Dc+F;
//矩形 1
   Parked_Car1x1=0;
   Parked_Car1y1=-6;
   Parked_Car1x2=0;
```

```
    Parked_Car1y2=22;
    Parked_Car1x3=15;
    Parked_Car1y3=22;
    Parked_Car1x4=15;
    Parked_Car1y4=-6;
//矩形 2
    Parked_Car2x1=0;
    Parked_Car2y1=70;
    Parked_Car2x2=0;
    Parked_Car2y2=100;
    Parked_Car2x3=15;
    Parked_Car2y3= 100;
    Parked_Car2x4=15;
    Parked_Car2y4=70;
//参数
    W=45;
    W=(W/180)*pi; % degree to radians
    S=3;
    xi=Parked_Car2x4+S;
    yi=Parked_Car2y4-9;
f1e=38;
模拟设置
    axis([-60 100 -20 100],'equal');
t=0:0.01:pi;
    seta=2*t;
    Xc=xi-(Lc/tan(W));
    Yc=yi+Dc;


    Rbl=sqrt(Dc^2+(Lc/tan(W))^2);
    phi=atan(Dc/(Lc/tan(W)));
//启动小车
          i=1;
          x=Xc+Rbl*cos(seta(i)+phi);
          y=Yc-Rbl*sin(seta(i)+phi);
          patch([Parked_Car2x1 Parked_Car2x2 Parked_Car2x3 Parked_Car2x
4],[Parked_Car2y1 Parked_Car2y2 Parked_Car2y3 Parked_Car2y4],[1 0 0]);
          patch([Parked_Car1x1 Parked_Car1x2 Parked_Car1x3 Parked_Car1x
4],[Parked_Car1y1 Parked_Car1y2 Parked_Car1y3 Parked_Car1y4],[0 0 1]);
          Turn1(x,y,A,B,seta(i));
pause(4);
//场景模拟
    for i=2:f1e
          x=Xc+Rbl*cos(seta(i)+phi);
```

```
            y=Yc-Rbl*sin(seta(i)+phi);
            cla;
            patch([Parked_Car2x1 Parked_Car2x2 Parked_Car2x3 Parked_Car2x
4],[Parked_Car2y1 Parked_Car2y2 Parked_Car2y3 Parked_Car2y4],[1 0 0]);
            patch([Parked_Car1x1 Parked_Car1x2 Parked_Car1x3 Parked_Car1x
4],[Parked_Car1y1 Parked_Car1y2 Parked_Car1y3 Parked_Car1y4],[0 0 1]);
            Turn1(x,y,A,B,seta(i));
            pause(.05);
       end
```

问题三：

```
import numpy as np
import matplotlib.pyplot as plt

plt.rcParams["font.sans-serif"] = ['SimHei']  # 用于正常显示中文标签
plt.rcParams['axes.unicode_minus'] = False  # 用来正常显示负号

x = np.linspace(0, 2, 300, endpoint=True)

# 0<w<1 概率
f1 = (1-0.1)*x*(0.1**x)  # y 值
f2 = (1-0.2)*x*(0.2**x)
f3 = (1-0.3)*x*(0.3**x)
f4 = (1-0.4)*x*(0.4**x)
f5 = (1-0.5)*x*(0.5**x)

plt.plot(x, f1, "black", lw=1.5, label="w=0.1")
plt.plot(x, f2, "yellow", lw=1.5, label="w=0.2")
plt.plot(x, f3, "green", lw=1.5, label="w=0.3")
plt.plot(x, f4, "red", lw=1.5, label="w=0.4")
plt.plot(x, f5, "blue", lw=1.5, label="w=0.5")

plt.title('f(x) =(1-w)×X×w^X 函数图')

plt.legend()  # 显示图例
plt.show()

import math
Smax=2.4*29+0.5 #直线行驶最远距离 29 个车位*每个车位 2.4m
Vmax= 20/3.6  #小车最大速度为 20km/h
Vj=10/3.6 #减速带允许的最大速度
```

```
#(x,y)小车坐标
#(xi,yi)车位坐标
L=1
Rmin=4.995

#车位坐标
# lis=[{'x':31.2,'y':0,'xi':64.8,'yi':0,'alpha':0,'beta':0},
# {'x':38.4,'y':16.2,'xi':64.8,'yi':0,'alpha':math.pi,'beta':math.pi/2},
# {'x':30.6,'y':5.5,'xi':64.8,'yi':0,'alpha':0,'beta':math.pi/2},
# {'x':35.7,'y':10.7,'xi':64.8,'yi':0,'alpha':math.pi,'beta':math.pi/2}]

def ki(alpha):    #弯道数计算
    if alpha==0:
        return 0
    elif abs(alpha - math.pi / 2 < 0.01):
        return 1
    elif abs(alpha-math.pi)<0.01:
        return 2

def Tm1(k):  #计算弯道路径时间
    k=ki(alpha)
    return (2*math.pi*Rmin*k)/Vmax

def S2(alpha,x,y,xi,yi):   #计算直线行驶距离
    delta_xi=x-xi
    delta_yi=y-yi
    if alpha==0:
        return abs(delta_xi)
    elif abs(alpha - math.pi / 2 < 0.01):
        return delta_xi+delta_yi
    elif abs(alpha - math.pi) < 0.01:
        return 2*Smax-(x+xi)

def Tm2(alpha,x,y,xi,yi):
    delta_xi = x - xi
    delta_yi = y - yi
    if alpha == 0:
        if Smax-x>5 and delta_xi>0:
            return (delta_xi)/Vmax
        elif Smax-x<=5 and delta_xi>0:
            return (Smax-x)/Vj+delta_xi/Vmax
        elif delta_xi<0:
            return -delta_xi/Vj
    elif abs(alpha-math.pi/2<0.01):
```

```
            if Smax-x>5 and delta_xi>0:
                return (Smax-x-5)/Vmax+(Smax-x+delta_yi)/Vj
            elif Smax-x<=5 and delta_xi>0:
                return (Smax-x+delta_yi)/Vj
        elif abs(alpha - math.pi) < 0.01:
            if Smax-xi>=5 and Smax-x>5:
                return (Smax+x-5)/Vmax+(Smax-x+delta_yi)/Vj+5/Vj+(Smax-xi-5)/V
max
            elif Smax-xi<=5 and Smax-x>5:
                (Smax+x-5)/Vmax+(Smax-x+delta_yi)/Vj+(Smax-xi)/Vj
            elif Smax-xi>5 and Smax-x<=5:
                return (Smax+x+delta_yi)/Vj+5/Vj+(Smax-xi-5)/Vj
            elif Smax-xi<=5 and Smax-x<=5:
                return (Smax+x+delta_yi)/Vj+5/Vj+(Smax-xi)/Vj

    def Tm3(beta):   #计算倒车时间
        return (Rmin*beta)/Vmax

    def Sum_T(alpha,beta,x,y,xi,yi):   #计算总时间
        k=ki(alpha)
        t1=Tm1(k)
        t2=Tm2(alpha,x,y,xi,yi)
        t3=Tm3(beta)
        sum=t1+t2+t3
        return sum

    while True:
        x,y,xi,yi,alpha,beta=map(eval,input().split())
        print(Sum_T(alpha,beta,x,y,xi,yi))
```

问题四：

```
import random
member_number =85
members = []
groups = []

# 创建列表
for i in range(1, member_number + 1):
    members.append(i)

#产生30个随机车位
for i in range (1,31):
    n=random.randint(1,85)
```

```
    groups.append(n)
#车位排序
groups.sort()
#输出车位
print(groups)

import random
member_number =85
members = []
groups = []
groups_spare=[]

# 创建列表
for i in range(1, member_number + 1):
    members.append(i)

#产生 30 个随机被占用车位
for i in range (1,31):
    n=random.randint(1,85)
    groups.append(n)

#空余车辆
groups_spare=members
for i in members:
    if i in groups:
        groups_spare.pop(i)

#车位排序
groups.sort()
groups_spare.sort()

#输出空余车位
print(groups_spare)

#随机离场一辆车
rin=random.randint(0,len(groups))
groups.pop(groups[rin])
groups_spare.append(groups[rin])

#随机进场一辆车
rout=random.randint(0,len(groups_spare))
groups_spare.pop(groups_spare[rout])
groups.append(groups_spare[rout])
```