自动驾驶中的转弯问题研究

摘要

当今时代科技飞速发展，自动驾驶技术已步入大众视野，但作为汽车行业最新的发展方向，这项技术仍存在一系列的问题待解决，模型预测控制是研究此类问题较为便捷的方法，本文将通过数学模型的建立针对自动驾驶在转弯方面的问题进行研究解决。由于无法得知汽车在实地情况下内外界因素对它的影响，所以本文将忽略这类因素，只考虑理想情况下汽车转向问题。

针对问题一，要对一辆前轮驱动的四轮车辆，初步建立描述车辆转弯的数学模型，由于车辆转弯所要考虑方面较多，计算程序较为复杂，复杂模型较简单模型更为精确，但对于自动驾驶技术来说，计算和处理的时间也是十分重要的考虑因素，综合考虑并结合本题要求可以简化除真实反映车辆特性外的不必要的因素，最终通过简单的单车模型以及车辆运动学模型进行数学模型的建立。我们针对建立的模型也进行了一系列的分析和可视化展示，综合起来得出结论，较低速度、较小前轮转角行驶的无人驾驶汽车具有更高的稳定性和安全性。

针对问题二，对比问题一我们得到了更多的车辆相关参数，所要考虑的因素也随之增多，为方便总结，首先对得到的数据进行整理加工，再将相关数据与问题一所构建的数学模型相结合，可以轻松构建出车辆转弯的模型。通过对于时间和步长的模拟，可视化地展现转弯的效果。

针对问题三，求车前内轮的最大和最小转弯角度，及车辆开始转弯的位置，此问的难点在于如何确定转动圆心和转动半径，也就是确定开始转弯的位置和转弯的角度。所建立的模型需要保证不与障碍物接触，可认为模型与障碍物刚好接触时为临界点，可将其简化为直线运动与简单圆周运动两个运动过程，通过对内后轮和外前轮的运动轨迹进行分析来解答。此外，最终通过蒙特卡罗法对于我们模型的合理性进行了统计意义上的检验。

本文设计的算法及模型对不同型号车辆均可进行自动驾驶转弯应用，具有普适性，所有算法以及建模代码均在附录给出。

关键词：自动驾驶；模型预测控制；转弯问题；单车模型；蒙特卡罗法

目录

1. 问题重述

1.1 问题背景

随着科学技术在各个领域的迅速发展，自动驾驶技术也正在慢慢走进人们的生活。自动驾驶技术承诺提供安全、舒适和高效的驾驶体验。然而，自动驾驶也面临着一系列的技术难题和安全挑战。

技术难题方面，自动驾驶汽车需要精确、实时地感知和理解环境，在遇到复杂天气时，更需要保持高精度的感知能力，以便做出高效的驾驶决策，图 1 展示了自动驾驶所需要的硬件和软件支持。

安全挑战方面，汽车自动驾驶技术也带来了一系列的安全问题，例如，2020 年 9 月 5 日一辆特斯拉 Model X 失控造成交通事故 2 死 6 伤，状况惨烈[1]。上述种种皆表明，自动驾驶技术在大规模投入实际应用仍然有很长的一段路，因而针对此问题，要保证自动驾驶汽车的快速、安全上路，不仅要依靠仿真测试，还需要提高自动驾驶汽车场景建模的准确性和数据分析的有效性[2]。

根据自动驾驶相关综述研究[3]，如果能够实现自动驾驶车辆的广泛部署，预计到 2050 年，自动驾驶的年度社会效益将达到近 8000 亿美元，通过重新分配驾驶时间来缓解拥堵、减少道路伤亡、降低能源消耗和提高生产率，然而无人驾驶技术作为不确定环境中运行的复杂机器人系统，有无数的场景存在未解决的问题。转弯控制对于自动驾驶这样一种对于安全性能要求极高的技术，就显得尤为重要，出现一丝一毫的失误就可能带来无法挽回的后果。正因如此，我国也需要快速发展可靠、安全的自动驾驶技术，本赛题正是在此背景下要解决自动驾驶中车辆转弯的问题。

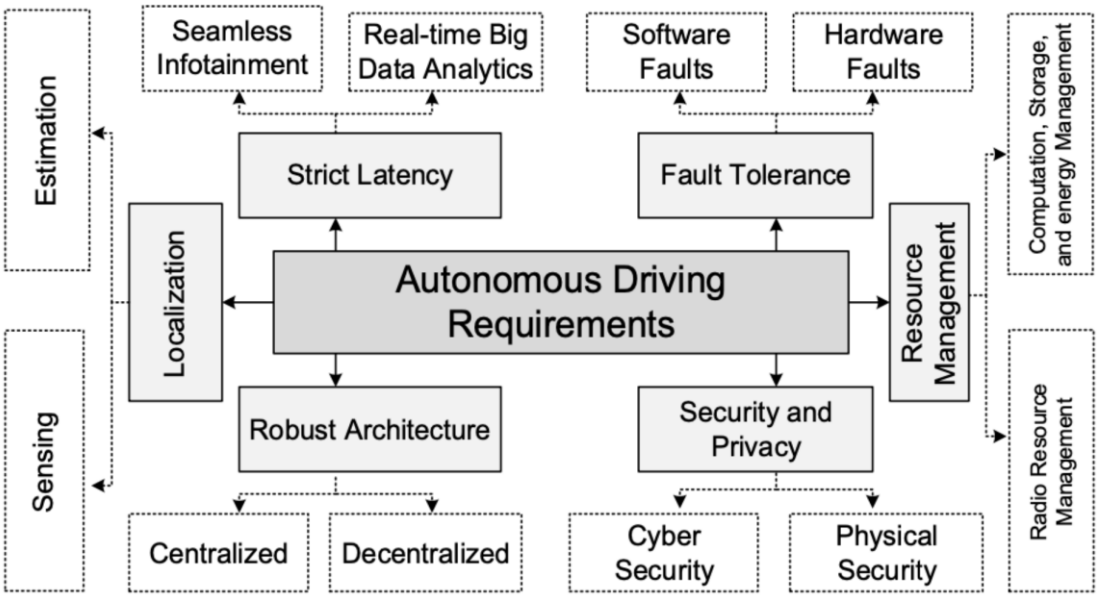

图 1：智能城市中的自动驾驶汽车：最新进展、要求和挑战

1.2 问题描述

基于上述的研究背景，题目提供了一些需要模拟的车辆数据，围绕自动驾

驶的转向问题，本文将解决以下三个问题：

问题 1：针对一辆前轮四驱车辆，初步建立描述车辆转弯的数学模型。这需要结合车辆动力学的相关知识来构建相对应的数学模型。

问题 2：根据问题 1 中建立的数学模型，给出每隔 0.1s 某一具体车辆的位置。

问题 3：考虑一个实际的应用场景，模拟车辆出库，确定该车前内轮的最大和最小转弯角度，及车辆开始转弯的位置。

2. 模型假设与符号说明

2.1 模型假设

参考相关资料，对于车辆模型进行了如下假设：

（1）忽略空气阻力等的影响。

（2）忽略悬架系统的作用，假设车身和悬架系统都是刚性系统。

（3）不考虑车辆在垂直方向的运动，假设车辆的运动只是存在于二维平面上的运动。

（4）假设车辆左右两侧轮胎在任意时刻都拥有相同的转向角度和转速。

（5）假设转弯过程车辆都已降速，以较低速度进行转向。

（6）假设车辆的运动和转向都是由前轮驱动。

以上假设针对本文中所要应对的问题，一定程度上是较为合理的。由于问题中并不知晓车辆的具体重量以及所受到的相互作用等种种因素，查阅相关资料，动力学模型需要考虑整车质量，通过牛顿力学关系建立，同时考虑了轮胎特性即轮胎的侧偏，不适合本文所要应对的场景。

2.2 符号说明

符号	符号说明
x	车辆当前的x坐标
y	车辆当前的y坐标
ψ	车辆当前的偏航角，即车辆与x轴的夹角
v	车辆当前的速度
δ_r	后轮偏角，此处默认为 0

δ_f	前轮偏角
β	滑移角，即车辆行进方向与前轮所指方向之间的角度
l_f	前半轴距
l_r	后半轴距
a	加速度

表 1：符号说明

3. 问题一分析与求解

3.1 问题分析

问题一对于一辆前轮驱动的四驱车辆建立描述车辆转弯的数学模型。自动驾驶汽车由复杂的软硬件系统构成，相应的，其状态受到诸多因素的影响，例如周围环境、天气状况、路面状况、传感器灵敏程度等。将其转化为数学模型并能够完整反映其运动特征是极其困难的事。

同时，复杂模型虽然会带来准确性上的提升，但会大大增加计算和处理的时间，对于自动驾驶这一对于实时性有较高要求的嵌入式系统，无疑是有很大影响的。对于像这样的嵌入式系统，计算的正确性不仅取决于程序的逻辑正确性，也取决于结果产生的时间，时间约束条件得不到满足就会出错。

因此，本文仅仅根据需要解决的实际问题，建立相对应的车辆运动模型，而忽略其他因素。在模型假设的基础上，我们将基于简单的自行车模型以及车辆运动学模型进行数学模型的建立。

3.2 模型建立

基于问题分析，我们建立的模型除了要真实反映车辆特性外，也应当尽量简单易用。鉴于此前的假设，转弯的速度应该是尽量低的，因此，我们采取了前轮驱动的单车运动模型，模型示意如图 2 所示。

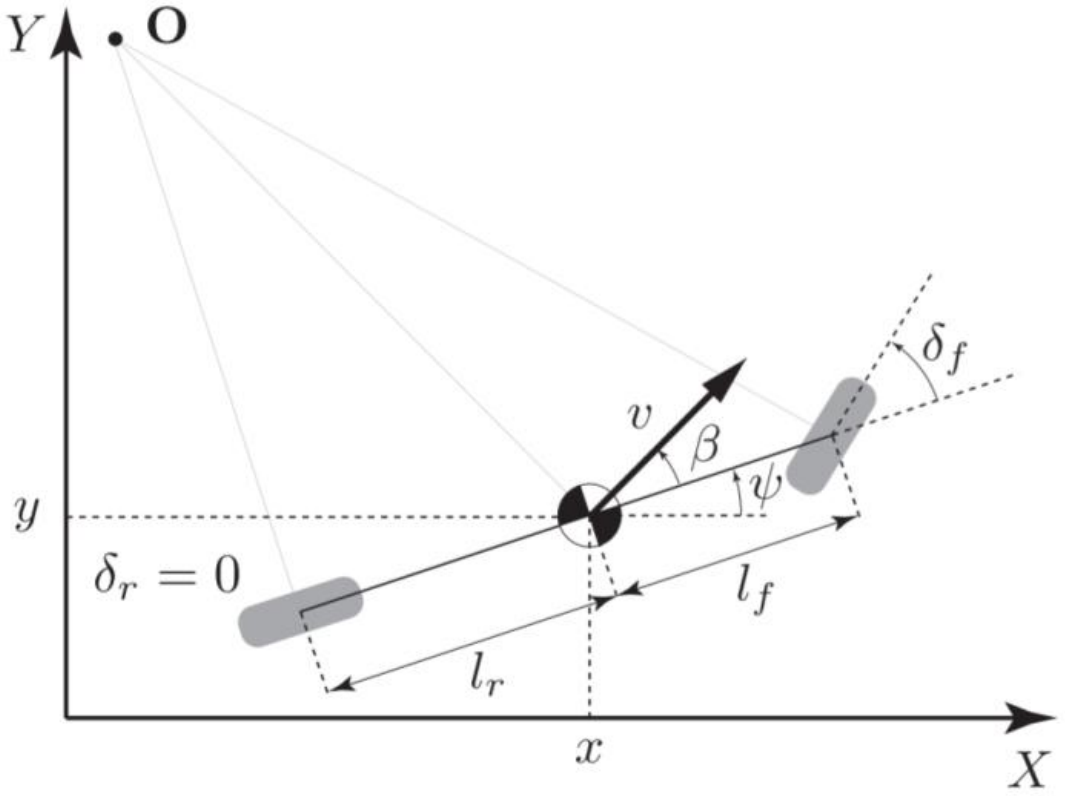

图 2：运动学自行车模型

针对上述模型，可以建立起以下离散化的计算公式：

图 2 中的车辆当前的偏航角用ψ_t表示，滑移角用β表示，$v_t\cos(\psi_t+\beta)\times dt$即为$x$在$dt$时刻后的变化量，$x$变化后的坐标为：

$$x_{t+1} = x_t + v_t\cos(\psi_t+\beta)\times dt$$

同样的，dt时刻后，y变化后的坐标为：

$$y_{t+1} = y_t + v_t\sin(\psi_t+\beta)\times dt$$

此时，车辆的偏航角为：

$$\psi_{t+1} = \psi_t + \frac{v_t}{l_r}\sin(\beta)\times dt$$

如果做变速运动，则此刻的速度为：

$$v_{t+1} = v_t + a\times dt$$

特别的，由于进行了特殊处理，将后轮偏角设置为 0，我们可以求得滑移角，即车辆行进方向与前轮所指方向之间的角度：

$$\beta = \tan^{-1}\left(\frac{l_r}{l_f+l_r}\tan(\delta_f)\right)$$

3.3 问题求解

使用本节构建的模型得到的前轮驱动单车运动模型的路径仿真如图 3、

4、5 所示。其中假设前半轴距与后半轴距相等，均为 1.25。其中，图 3 为速度为 10m/s，前轮偏角δ_f为 10 度的情况；图 4 为速度为 5m/s，前轮偏角δ_f为 10 度的情况；图 4 为速度为 10m/s，前轮偏角δ_f为 20 度的情况。

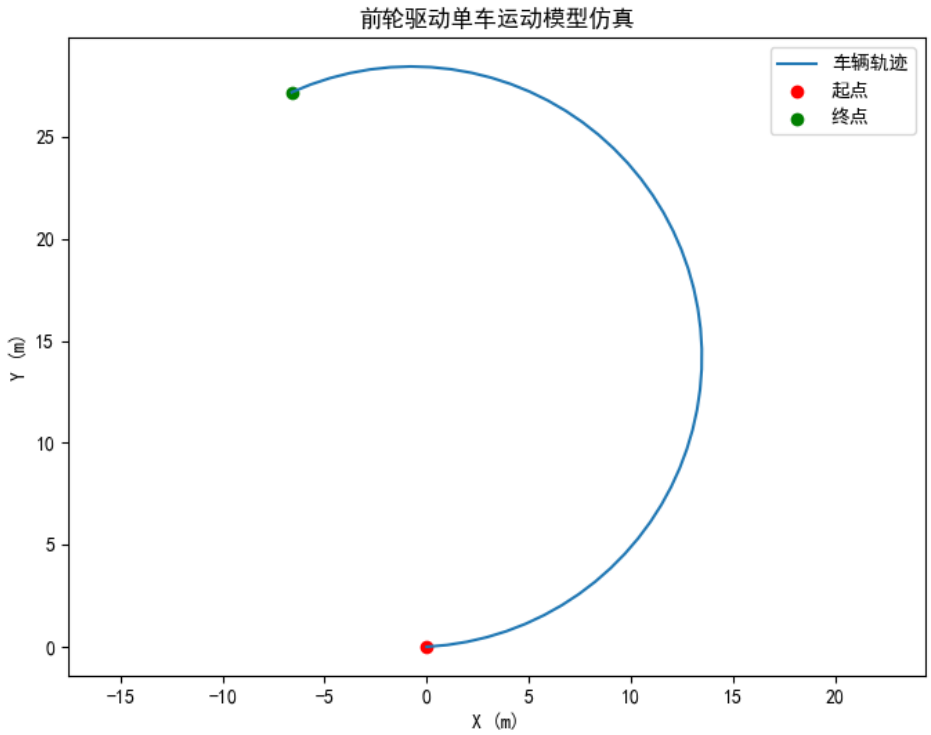

图 3：速度为 10m/s，前轮偏角为 10 度车辆轨迹

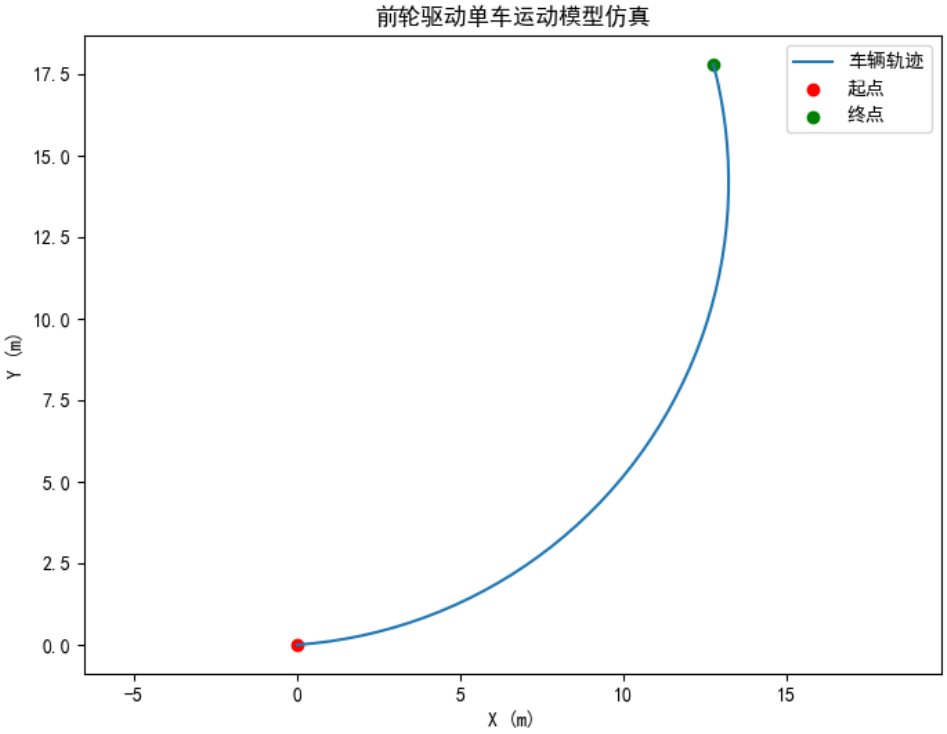

图 4：速度为 5m/s，前轮偏角为 10 度车辆轨迹

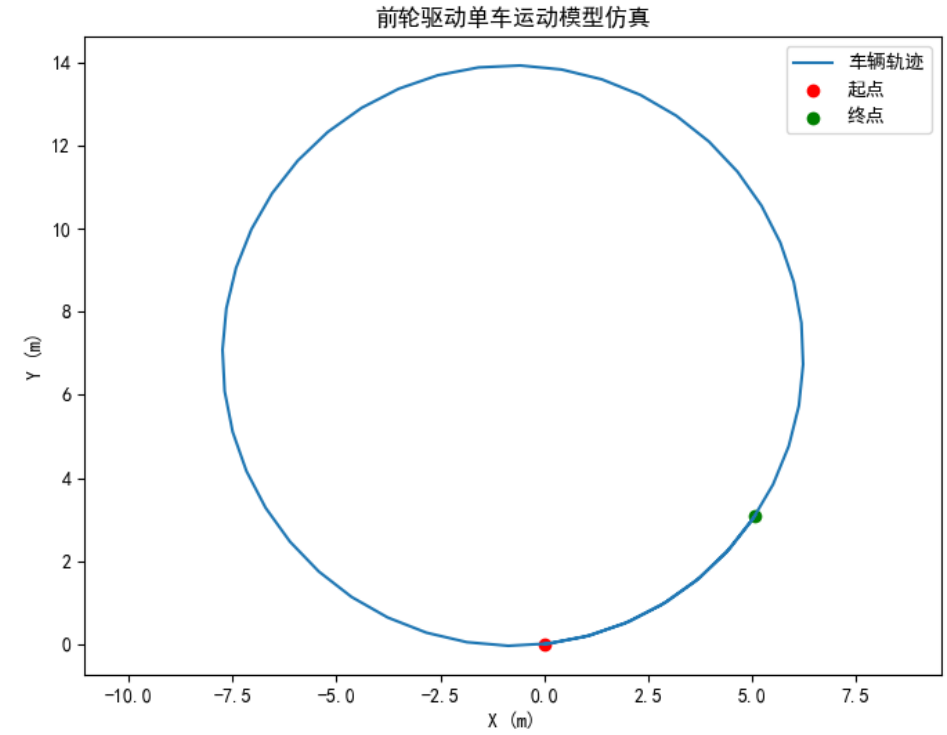

图 5：速度为 10m/s，前轮偏角为 20 度车辆轨迹

对于该模型进一步分析，比较相同速度、不同前轮偏角下车辆前轮的横向

位移，如图 6 所示：

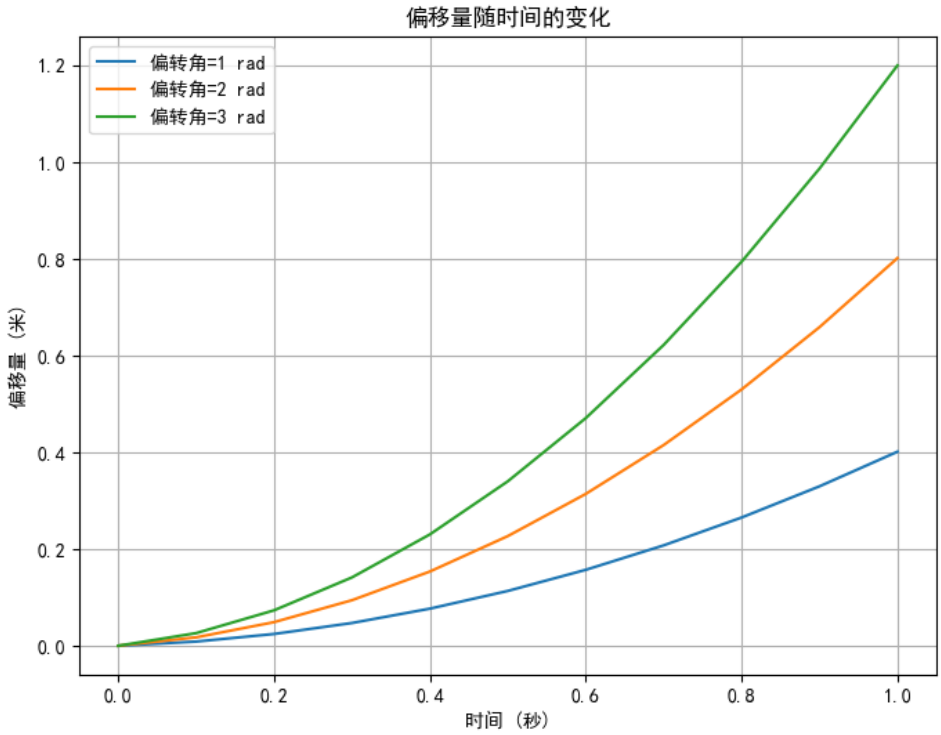

图 6：不同前轮偏角下前轮水平偏移量

由上图结果可知，随着前轮偏角（即假定中的方向盘旋转角度）增大，前轮水平偏移量增大且随时间增长，偏移速度变快，由此可以发现，直线行驶时应当避免方向盘较大幅度的摆动，避免偏离原本行驶路线过多。

同样的，我们可以分析相同前轮偏角下该自动驾驶转弯模型在不同转弯速度下的曲线，如图 7 所示：

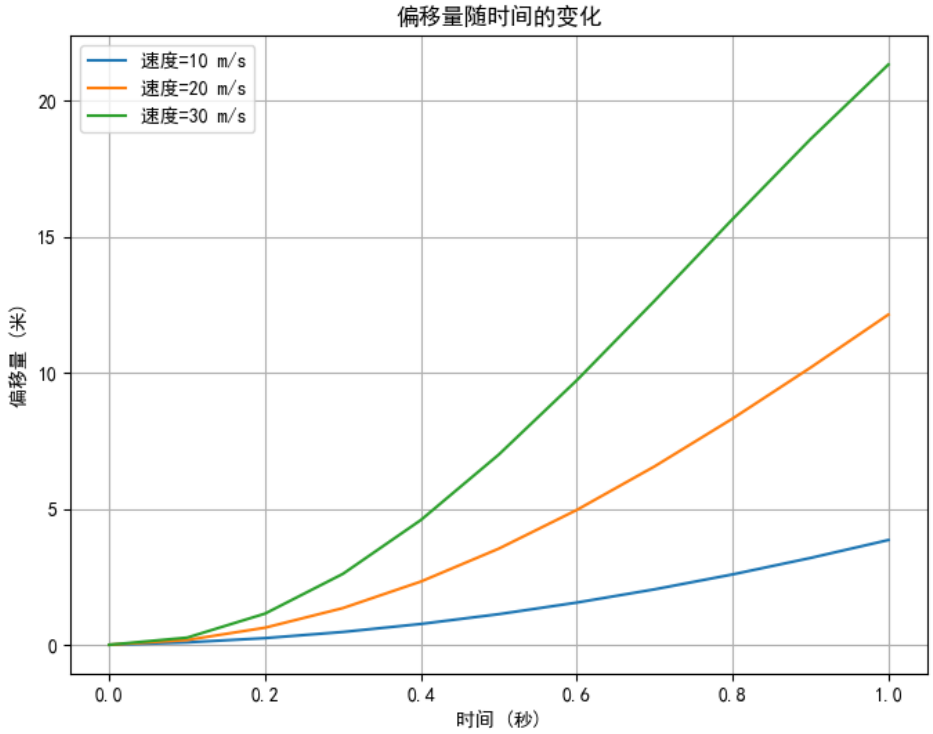

图 7：不同速度下前轮水平偏移量

由上图结果可知，随着速度增大，同一时刻前轮水平偏移量增大，偏移速度变快，由此可以发现，应避免较高速度转向从而造成较大的水平偏移。因此，较低的行驶速度可以使汽车具有更好的操纵性。

由上述综合起来可以得出结论，较低速度、较小前轮转角行驶的无人驾驶汽车具有更高的稳定性和安全性。

4. 问题二分析与求解

4.1 问题分析

4.1.1 数据处理

问题二给出了车辆的相关参数，并要求根据问题一的数学模型给出每隔 0.1 秒车辆的位置，相较于第一问，我们所得到的参数更多，所要考虑的方面更多，因此，我们首先整理问题 2 中给出的相关数据：车辆长度 $L = 4m$，宽度 $W = 2m$，车轮直径 $d = 0.6m$，宽度 $w = 0.16m$，前轮到车头距离 $a = 0.5m$，后轮到车尾距离 $b = 0.5m$，车辆转弯速度为 $V = 20km/h$，前内轮的转弯角度 $\delta_f = 30°$。

为了方便计算，我们应将速度和转弯角度的单位换算，换算得 $V = 5.56m/s$

车辆中心在转弯开始时的位置： $x = y = 0$

假定前梁后梁长度相等，取前后轮胎中心之间的距离为梁和，各 1/2 为前后梁的长度，即有：

$$l_f = l_r = (4 - 0.5 * 2 - 0.6)/2 = 1.2m$$

现在根据所整理的数据进行分析，思考模型的建立。

4.1.2 分析问题

借助于问题 1 的数学模型，我们可以很轻松的构建出车辆转弯的模型，只需将上述处理好的数据带入问题 1 构建好的数学模型之中去。同时，我们可以很轻松的观察到前后车轮的中心与车辆整体的中心存在对应关系，确定车辆中心后，根据相关角度和长度计算即可计算出其余四个点的横纵坐标。

4.2 模型建立

基于上述分析，我们建立了问题二的数学模型。

具体来讲，其余四点到中心的距离在我们的假设，即中心为整车长宽的中心下，该距离公式如下：

$$l_{center} = \sqrt{\left(l_f^2 + (W - w)/2\right)^2}$$

四点与中心连线与车身方向的夹角满足以下公式：

$$\theta = arctan\left(((W - w)/2)/l_f\right)$$

以内前轮为例，其对应坐标计算公式为：

$$X = x + cos(\theta + \psi) * l_{center}$$
$$Y = y + sin(\theta + \psi) + l_{center}$$

4.3 问题求解

选取时间为 2s，对于该车辆转弯时间和步长进行模拟，得到如图 8 所示的车辆位置。将同一时刻的前后车轮中心四个点相连，即可构建描绘出该时刻车辆的位置信息，中心此刻的位置也大致在四个点的连线交点上。需注意以开始转弯时车辆中心为原点，车头方向为 y 轴。

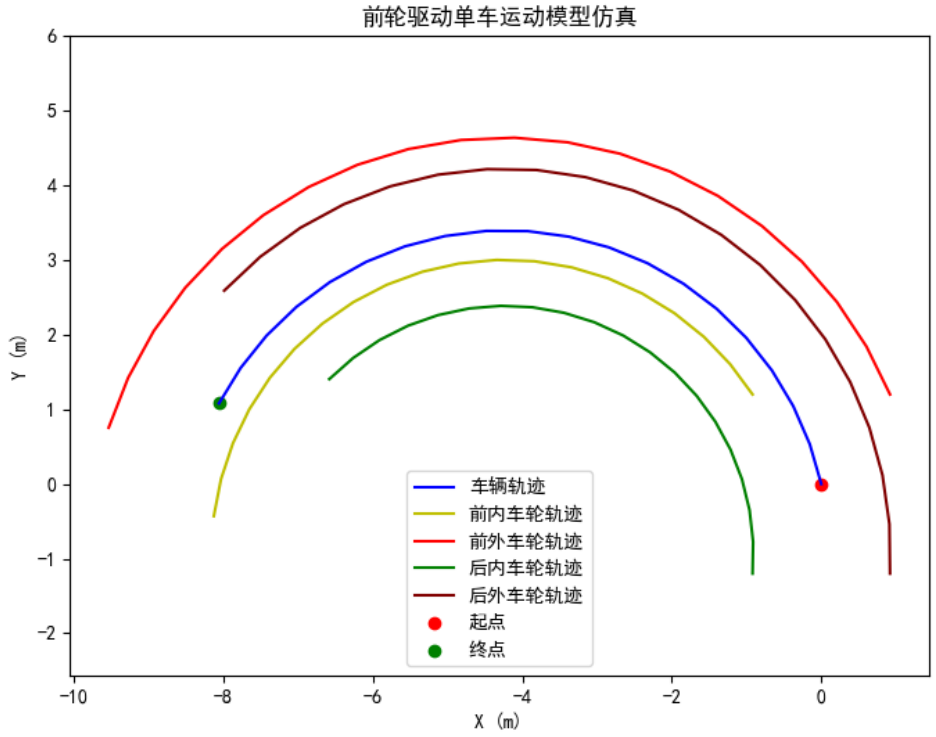

图 8：车辆位置

所记录完整数据如下表所示：

0	0	0	-0.92	1.2	0.92	1.2	-0.92	-1.2	0.92	-1.2
0.1	-0.15421	0.534188	-1.2204	1.606393	0.604425	1.842193	-0.91284	-0.77382	0.91199	-0.53802
0.2	-0.3756	1.044208	-1.57041	1.970938	0.209153	2.438649	-0.96035	-0.35023	0.819212	0.117479
0.3	-0.66053	1.521652	-1.96425	2.287622	-0.2593	2.979532	-1.06176	0.063772	0.643195	0.755682
0.4	-1.00429	1.958645	-2.39542	2.551225	-0.7932	3.455923	-1.21538	0.461367	0.386843	1.366065
0.5	-1.40122	2.347981	-2.85682	2.757398	-1.38375	3.859964	-1.41869	0.835997	0.054382	1.938564
0.6	-1.84477	2.683239	-3.34084	2.902741	-2.02121	4.184994	-1.66834	1.181483	-0.3487	2.463736
0.7	-2.32763	2.95889	-3.83949	2.984859	-2.69506	4.425652	-1.9602	1.492128	-0.81577	2.932922
0.8	-2.84183	3.170389	-4.34456	3.002395	-3.39421	4.57797	-2.28946	1.762809	-1.33911	3.338383
0.9	-3.3789	3.314249	-4.8477	2.955062	-4.1071	4.639435	-2.6507	1.989062	-1.9101	3.673435
1	-3.92997	3.388095	-5.34064	2.84364	-4.822	4.609034	-3.03795	2.167156	-2.51931	3.93255
1.1	-4.48597	3.390712	-5.81523	2.669967	-5.5271	4.487269	-3.44483	2.294155	-3.15671	4.111457
1.2	-5.03771	3.322055	-6.26364	2.436906	-6.21079	4.276146	-3.86463	2.367964	-3.81178	4.207204
1.3	-5.57611	3.183257	-6.6785	2.1483	-6.86178	3.979149	-4.29044	2.387365	-4.47372	4.218214
1.4	-6.09228	2.976607	-7.05295	1.80891	-7.46935	3.601174	-4.71521	2.352039	-5.13161	4.144303
1.5	-6.57771	2.705512	-7.38081	1.424332	-8.02346	3.148456	-5.13196	2.262568	-5.77461	3.986692
1.6	-7.0244	2.374443	-7.65669	1.000908	-8.51499	2.628459	-5.5338	2.120428	-6.3921	3.747979
1.7	-7.42497	1.988861	-7.87604	0.54562	-8.93583	2.049758	-5.91412	1.927963	-6.97391	3.432101
1.8	-7.77284	1.555123	-8.03522	0.065978	-9.27904	1.421898	-6.26663	1.688347	-7.51045	3.044268
1.9	-8.06224	1.080382	-8.13163	-0.43011	-9.53896	0.755233	-6.58553	1.405531	-7.99286	2.590874

5. 问题三分析与求解

5.1 问题分析

问题三给出了停车位的相关参数，车辆停放在停车位的位置，道路的宽度以及车辆周围的环境情况。在确定车辆 A 左转出库，并且出库过程中不允许车

辆倒车的限定条件下，要求我们确定该车前内轮的最大和最小转弯角度，及车辆开始转弯的位置。

首先，车辆 A 需要左转出库，因此我们需要考虑车辆的转弯半径以及周围障碍物的距离。最大转弯角度将在车辆的内轮开始与周围障碍物相切时达到。因此，我们需要计算车辆的内轮与周围障碍物的最大接触点。最小转弯角度将在车辆的外轮开始与周围障碍物相切时达到。同样，我们需要计算车辆的外轮与周围障碍物的最小接触点。其次要确定开始转弯的位置，车辆开始转弯的位置将取决于车辆与停车位的相对位置，以及车辆转弯的限制条件。

5.2 模型建立

基于上述分析，不难发现，本题的难点在于如何确定转动圆心和转动半径，也就是确定开始转弯的位置和转弯的角度。所建立的模型需要保证不与障碍物接触，不禁让人想到 pid 算法控制汽车运动的嵌入式小实验。其基本原理是通过激光雷达获取激光扫描距离，计算所需的转向角度和速度，以此来驱动汽车。本题并非是需要我们实时进行判断是否将要触碰到墙壁并保持某种约束条件不断调整车身姿态。因此，我们对于模型进行简化，将其简化为两个运动过程：1.直线运动；2.简单圆周运动。我们沿用问题二对于车辆一系列参数的规定，忽略前内轮转向角度的规定。由于问题三并未对速度进行规定，但作为停车场中出库这一场景，考虑实际情况，我们认为出库速度理应是极其小的。

同时，我们规定只能以恒定的转弯角度和速度进行转弯操作，不允许车辆倒车，也就是出现偏差不允许进行修正。

基于上述的一系列规定和假设，尤其是出库速度极其缓慢的情况下，在阿克曼转向几何中，低速环境下，车辆行驶路径半径变化缓慢。此时我们以后轴中心为参考点，近似以该点得出的圆心作为整体的圆心，可以假设车辆的方向变化率等于车辆的角速度，由此可以进一步推导近似出半径。推导过程如下：

根据角度的关系，利用$tan\delta = L/R$，得出$R = L/tan\delta$。

要是更进一步的，由于δ是较小的，我们可以近似L代替弧长，由此根据弧长与角度关系，可以得到$L/R = \delta$，即$R = L/\delta$。为了体现四个轮子半径的不等，我们还是使用上面的。

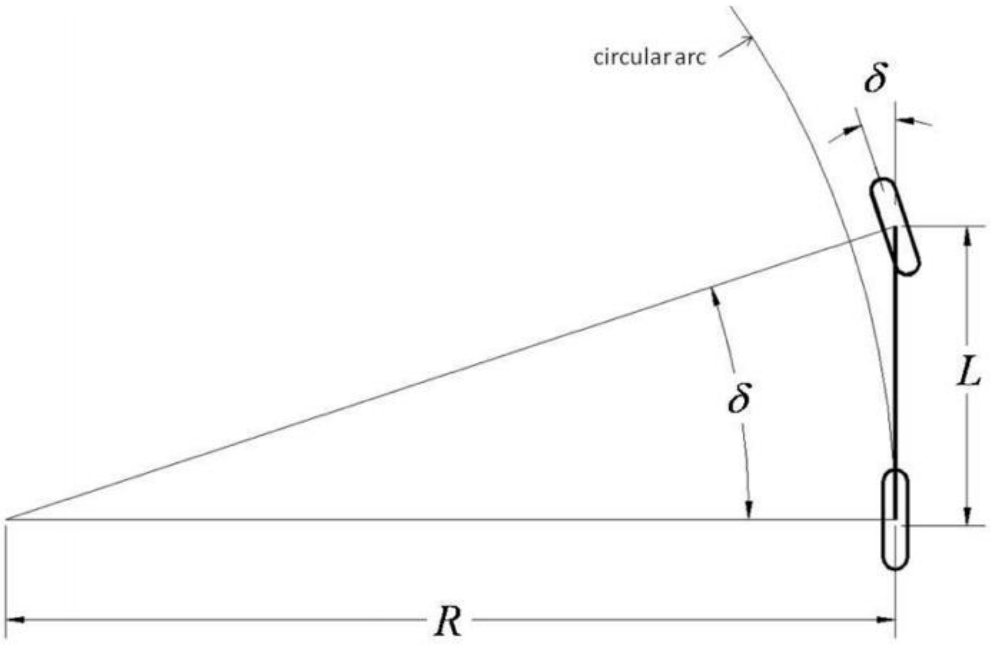

图 9：简化后的转弯模型

故，本题中$R = 2.4/tan\delta_f$，接下来就是圆相切的问题了。

同时，根据问题二画出的仿真示意图，我们不难发现，内后轮运动半径最小，外前轮运动半径最大，也正如图 10 所示。考虑内后轮和外前轮的运动轨迹：

$$R_{内后} = R - W/2 + w/2 = R - 0.92$$

$$R_{外前} = \sqrt{(R + W/2 - w/2)^2 + L^2} = \sqrt{(R + 0.92)^2 + 2.4^2}$$

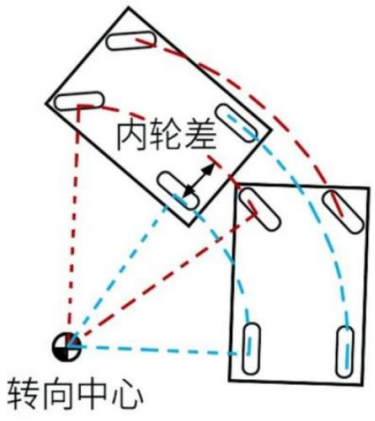

图 10：转向示意图

5.3 问题求解

5.3.1 内后轮

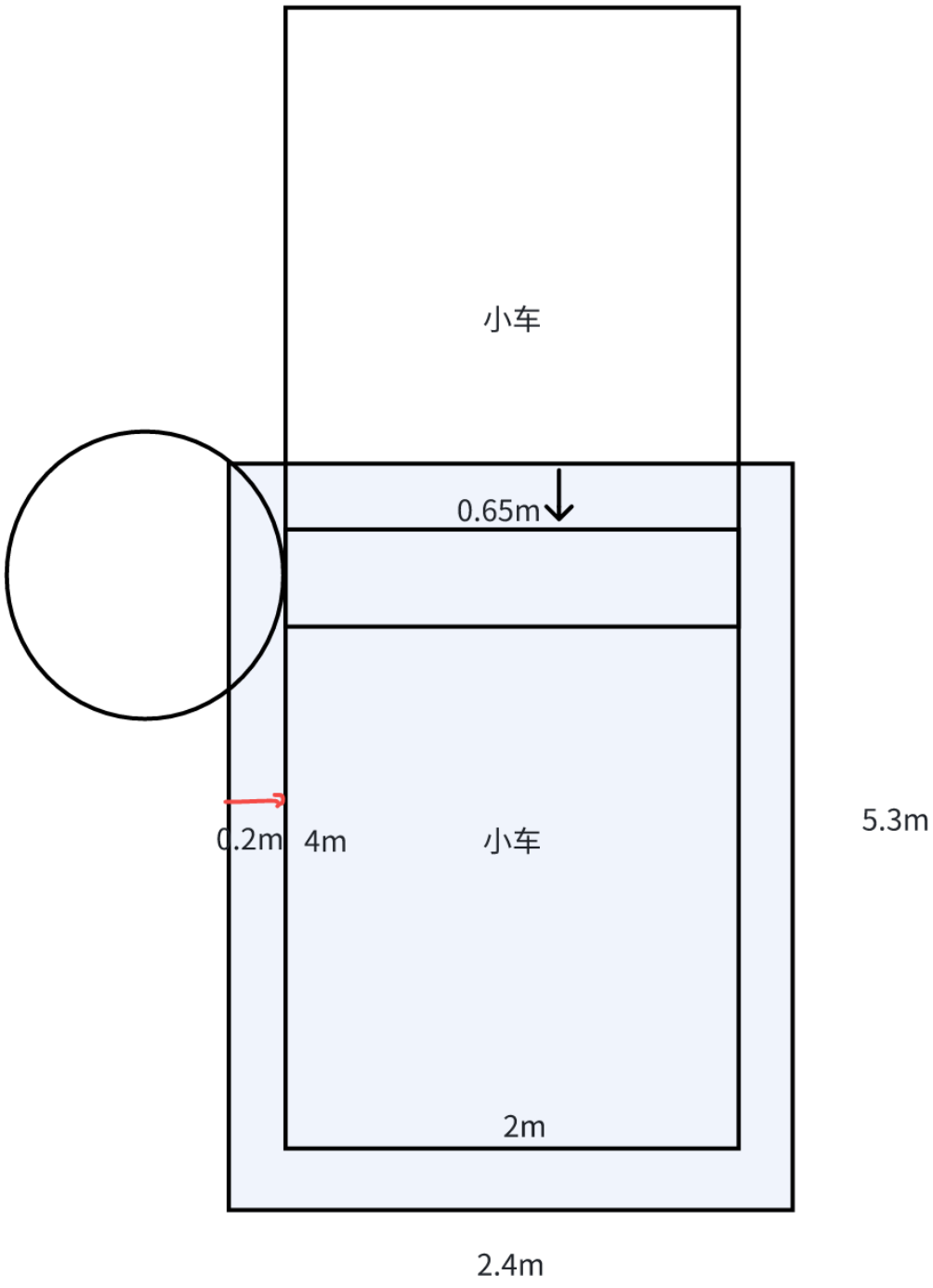

图 11：内后轮出库示意

可以构建出一个极其简单的方程：

$$R_{内后}^2 = \left(R_{内后} - 0.2\right)^2 + a^2$$

方程满足$R_{内后} >= 0.2$，带入解得$R_{min} = 1.12$，此时$\delta = 64.98°$，开始转弯位置位于后轮中心距离出车库线 0.2m 处。

查阅相关资料获知，汽车前轮转角的大小会因车辆的实际尺寸和运载底盘的不同而有所差异。一般来说，汽车的转向角度通常在 30 度到 40 度之间。例如，面包车的转向角大多在 30～34 度之间，而轿车和 SUV 等车型的转向角则大多为 40 度。故而取$\delta = 40°$，此时$R_{内后} = 1.94\text{m}$，开始转弯位置位于后轮中心距离出车库线 0.8579m 处。

5.3.2 外前轮

不进行直线运动，以一个小的角度直接做转弯，此刻我们可以认为转角是最小的，满足以下方程：

$$R_{外前} = 5.5 + 5.3 - 0.65 - 0.5 - 0.6/2 = 9.35m$$

带入解得$\delta = 16.47°$，此时开始转弯位置就位于原位置。

5.3.3 检验

上述问题求解我们使用了数学的思想，依据约束关系去求解角度和转弯位置。如何对于我们的结果进行检验呢？我们想到了蒙特卡罗法，基于随机抽样，根据统计试验求近似解。使用蒙特卡罗法得出的结果如图 12 所示。

```
前内轮最大转弯角度（角度）：  39.99964223998833
前内轮最小转弯角度（角度）：  16.50754670428598
最大转弯开始位置距离出库线（m）：  0.8579602331786038
最小转弯开始位置距离起点（m）：  0.017765266858285854
```

图 12：蒙特卡罗法检验

可以看出，统计意义上的最大转弯角度为 39.9996°，与我们的结果 40 近乎相等；最小转弯角度为 16.5075°，与我们的计算结果 16.47° 也是足够接近。最大转弯开始位置距离出库线的距离也近乎于 0.8579m，最小转弯开始位置距离起点位置为 0.018m，也是极其接近于 0 了。

因此，我们可以说，在统计意义上也验证了我们之前的模型构建分析是合理的。

6. 模型评价

6.1 模型的优点及创新

6.1.1 创新

1. 我们采用了自行车模型，以其高度简化的结构和直观的物理概念，成为了自动驾驶技术研究初期的理想工具。该模型通过将复杂的车辆运动特征抽象成为一个可操控的点质量与一对理想化的前后轮，极大降低了分析与实现的复杂度。这一简化不仅能使我们能够快速搭建起基础的仿真环境，也有助于我们理

解车辆运动的基本原理。

2. 我们还采取了蒙特卡罗法的统计思想对于我们的结果进行检验，证明构建模型具有较为科学的意义。

6.1.2 优点

1. 直观性:模型将复杂的车辆运动简化为几个关键参数（如车辆位置、航向角、速度和转向角）的相互作用，使得路径规划算法的设计与验证变得直接明了。

2. 低计算复杂度:相比多体动力学仿真，该模型的数学表达简洁，计算资源消耗小，有利于在算法开发初期快速迭代和测试不同控制策略的效果。

3. 易于实现控制逻辑:由于模型聚焦于车辆整体动态而非单个部件，控制策略的设计可以更加集中于全局路径跟踪和避障策略，有助于初探高级自动驾驶功能的基础逻辑。

6.2 模型的缺点及改进

6.2.1 缺点

1. 物理现实性的缺失:模型忽略了车辆的实际重量分布、悬挂系统动态响应等因素，这在处理紧急避障、复杂路况适应或高动态驾驶场景时，可能导致仿真结果与实际情况存在较大偏差。

2. 轮胎动力学简化:自行车模型没有考虑轮胎与地面之间复杂的相互作用，如轮胎侧偏特性、滑移率影响等，这些因素在车辆操控稳定性、制动性能评估中至关重要，其简化处理会限制对车辆极限状态的准确预测。

3. 环境交互的简化:在自行车模型中，通常不考虑风阻、坡道效应、路面附着系数变化等外部环境因素，这些在实际驾驶中会对车辆性能和操控产生显著影响，尤其是在设计针对特定环境的自动驾驶策略时。

6.2.2 改进

1. 多体动力学模型：考虑车辆的多个质量块及其间的相互作用，包括悬架系统、轮胎与地面的非线性接触，这能更准确地模拟车辆在各种工况下的动态响应。

2. 轮胎模型升级：采用更先进的轮胎模型，这些模型能更好地描述轮胎侧偏特性和滑动特性，增强车辆在不同路面条件下的仿真精度。

3. 环境交互的增强：传感器与环境融合，我们在上述中也提及了一些传感器交互的概念。在模型中集成雷达、摄像头等传感器数据处理逻辑，模拟自动驾驶系统如何根据周围环境动态调整行驶策略。

通过上述改进措施，不仅可以提升自动驾驶仿真模型的逼真度和预测准确性，还能增强自动驾驶系统在复杂、多变环境下的适应能力和安全性，推动自动驾驶技术向更高水平迈进。

参考文献

[1] Tesla accident (in Chinese). https://36kr.com/p/871536866489736

[2] 张梦寒, 杜德慧, 张铭茁, 张雷, 王耀, 周文韬. 时空轨迹数据驱动的自动驾驶场景元建模方法. 软件学报, 2021, 32(4): 973-987. http://www.jos.org.cn/1000-9825/6226.htm

[3] E. Yurtsever, J. Lambert, A. Carballo and K. Takeda, "A Survey of Autonomous Driving: Common Practices and Emerging Technologies," in IEEE Access, vol. 8, pp. 58443-58469, 2020, doi: 10.1109/ACCESS.2020.2983149.

附录

1. 前轮驱动单车运动模型

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

plt.rcParams['font.sans-serif'] = ['SimHei']  # 设置字体为黑体
plt.rcParams['axes.unicode_minus'] = False  # 用来正常显示负号

# 车辆参数
L_f = 1.25
L_r = 1.25
V = 10  # 车辆速度，单位：米/秒
delta = np.deg2rad(20)  # 前轮偏转角，单位：弧度

# 运动参数
dt = 0.1  # 时间间隔，单位：秒
total_time = 5  # 总仿真时间，单位：秒
steps = int(total_time / dt)  # 总步数

# 初始化车辆状态
x = 0  # 车辆 x 坐标，单位：米
y = 0  # 车辆 y 坐标，单位：米
psi = 0  # 车辆航向角，单位：弧度
beta = np.arctan((L_r / (L_r + L_f)) * np.tan(delta))


# 车辆运动轨迹
x_traj = [x]
```

```
y_traj = [y]

for _ in range(steps):
    # 更新 x、y 坐标以及与 x 轴夹角
    x += V * np.cos(psi + beta) * dt
    y += V * np.sin(psi + beta) * dt
    print(psi + beta)
    psi += (V / L_r) * np.sin(beta) * dt

    x_traj.append(x)
    y_traj.append(y)

# 绘制车辆轨迹
plt.figure(figsize=(8, 6))
plt.plot(x_traj, y_traj, label='车辆轨迹')
plt.scatter(x_traj[0], y_traj[0], color='r', marker='o', label='起点')
plt.scatter(x_traj[-1], y_traj[-1], color='g', marker='o', label='终点')
plt.axis('equal')
plt.xlabel('X (m)')
plt.ylabel('Y (m)')
plt.title('前轮驱动单车运动模型仿真')
plt.legend()
plt.show()
```

2. 不同前轮偏角车辆前轮水平偏移

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

plt.rcParams['font.sans-serif'] = ['SimHei']  # 设置字体为黑体
plt.rcParams['axes.unicode_minus'] = False  # 用来正常显示负号

# 车辆参数
L_f = 1.25
L_r = 1.25
V = 10  # 车辆速度，单位：米/秒
omega_values = [1, 2, 3]  # 偏转角，单位：弧度

# 运动参数
```

```
dt = 0.1  # 时间间隔，单位：秒
total_time = 1  # 总仿真时间，单位：秒
steps = int(total_time / dt)  # 总步数

# 绘制偏移量随时间的变化曲线
plt.figure(figsize=(8, 6))

for omega in omega_values:
    delta = np.deg2rad(omega)  # 前轮偏转角，单位：弧度
    # 初始化车辆状态
    x = 0  # 车辆 x 坐标，单位：米
    y = 0  # 车辆 y 坐标，单位：米
    psi = 0  # 车辆航向角，单位：弧度
    beta = np.arctan((L_r / (L_r + L_f)) * np.tan(delta))

    # 车辆运动轨迹
    x_traj = [x]
    y_traj = [y]

    for _ in range(steps):
        # 更新 x、y 坐标以及与 x 轴夹角
        x += V * np.cos(psi + beta) * dt
        y += V * np.sin(psi + beta) * dt
        psi += (V / L_r) * np.sin(beta) * dt

        x_traj.append(x)
        y_traj.append(y)

    t = np.linspace(0, total_time, steps + 1)
    plt.plot(t, y_traj, label=f'偏转角={omega} rad')

plt.xlabel('时间 (秒)')
plt.ylabel('偏移量 (米)')
plt.title('偏移量随时间的变化')
plt.legend()
plt.grid(True)
plt.show()
```

3. 不同速度下前轮水平偏移

```
Python
import numpy as np
```

```
import matplotlib.pyplot as plt

plt.rcParams['font.sans-serif'] = ['SimHei']  # 设置字体为黑体
plt.rcParams['axes.unicode_minus'] = False  # 用来正常显示负号

# 车辆参数
L_f = 1.25
L_r = 1.25
V = 10  # 车辆速度，单位：米/秒
delta = np.deg2rad(10)  # 前轮偏转角，单位：弧度
velocity_values = [10, 20, 30]  # 角速度，单位：弧度/秒

# 运动参数
dt = 0.1  # 时间间隔，单位：秒
total_time = 1  # 总仿真时间，单位：秒
steps = int(total_time / dt)  # 总步数

# 绘制偏移量随时间的变化曲线
plt.figure(figsize=(8, 6))

for velocity in velocity_values:
    # 初始化车辆状态
    x = 0  # 车辆 x 坐标，单位：米
    y = 0  # 车辆 y 坐标，单位：米
    psi = 0  # 车辆航向角，单位：弧度
    beta = np.arctan((L_r / (L_r + L_f)) * np.tan(delta))

    # 车辆运动轨迹
    x_traj = [x]
    y_traj = [y]
    V = velocity

    for _ in range(steps):
        # 更新 x、y 坐标以及与 x 轴夹角
        x += V * np.cos(psi + beta) * dt
        y += V * np.sin(psi + beta) * dt
        psi += (V / L_r) * np.sin(beta) * dt

        x_traj.append(x)
        y_traj.append(y)

    t = np.linspace(0, total_time, steps + 1)
```

```
    plt.plot(t, y_traj, label=f'速度={velocity} m/s')

plt.xlabel('时间 (秒)')
plt.ylabel('偏移量 (米)')
plt.title('偏移量随时间的变化')
plt.legend()
plt.grid(True)
plt.show()
```

4. 车辆转弯位置记录

```
Python
import numpy as np
import matplotlib.pyplot as plt
import math
import pandas as pd

plt.rcParams['font.sans-serif'] = ['SimHei']  # 设置字体为黑体
plt.rcParams['axes.unicode_minus'] = False  # 用来正常显示负号

# 车辆参数
L_f = 1.2
L_r = 1.2
V = 5.56  # 车辆速度，单位：米/秒
delta = np.deg2rad(30)  # 前轮偏转角，单位：弧度
car_length = 4.0  # 车身长度 m
car_width = 2.0   # 车身宽度 m
wheel_diameter = 0.6  # 车轮直径 m
wheel_width = 0.16  # 车轮宽度 m
front_offset = 0.5  # 前轮到车头距离 m
rear_offset = 0.5  # 后轮到车尾距离 m
# 轮子相对于车辆中心的横向和纵向偏移量
front_wheel_offset = car_length / 2 - front_offset -
wheel_diameter / 2
rear_wheel_offset = -car_length / 2 + rear_offset / 2 +
wheel_diameter / 2
side_wheel_offset = car_width / 2 - wheel_width / 2


len_center = math.sqrt(front_wheel_offset * front_wheel_offset +
side_wheel_offset * side_wheel_offset)
```

```
theta_rad = np.arctan(side_wheel_offset / front_wheel_offset)


# 运动参数
dt = 0.1  # 时间间隔，单位：秒
total_time = 2  # 总仿真时间，单位：秒
steps = int(total_time / dt)  # 总步数

# 初始化车辆状态
x = 0  # 车辆 x 坐标，单位：米
y = 0  # 车辆 y 坐标，单位：米
psi = np.deg2rad(90)  # 车辆航向角，单位：弧度，车头指向 y 轴
beta = np.arctan((L_r / (L_r + L_f)) * np.tan(delta))

# 车辆运动轨迹
x_traj = []
y_traj = []
x_front_in_traj = []
y_front_in_traj = []
x_front_out_traj = []
y_front_out_traj = []
x_rear_in_traj = []
y_rear_in_traj = []
x_rear_out_traj = []
y_rear_out_traj = []

# 初始化位置数据列表，包含车辆中心及四个轮子的位置
data = []

for step in range(steps):
    # 车辆中心位置
    x_center = x
    y_center = y


    # 各轮子位置计算
    x_front_in = x_center + np.cos(theta_rad + psi) * len_center
    y_front_in = y_center + np.sin(theta_rad + psi) * len_center

    x_front_out = x_center + np.cos(theta_rad - psi) * len_center
    y_front_out = y_center - np.sin(theta_rad - psi) * len_center
```

```
    x_rear_in = x_center - np.cos(theta_rad - psi) * len_center
    y_rear_in = y_center + np.sin(theta_rad - psi) * len_center

    x_rear_out = x_center - np.cos(theta_rad + psi) * len_center
    y_rear_out = y_center - np.sin(theta_rad + psi) * len_center

    # 收集当前时间步的数据
    data.append([step * dt,
                 x_center, y_center,
                 x_front_in, y_front_in,
                 x_front_out, y_front_out,
                 x_rear_in, y_rear_in,
                 x_rear_out, y_rear_out])


    x_traj.append(x)
    y_traj.append(y)

    x_front_in_traj.append(x_front_in)
    y_front_in_traj.append(y_front_in)
    x_front_out_traj.append(x_front_out)
    y_front_out_traj.append(y_front_out)
    x_rear_in_traj.append(x_rear_in)
    y_rear_in_traj.append(y_rear_in)
    x_rear_out_traj.append(x_rear_out)
    y_rear_out_traj.append(y_rear_out)


    # 更新 x、y 坐标以及与 x 轴夹角
    x += V * np.cos(psi + beta) * dt
    y += V * np.sin(psi + beta) * dt
    psi += (V / L_r) * np.sin(beta) * dt


# 绘制车辆轨迹
plt.figure(figsize=(8, 6))
plt.plot(x_traj, y_traj, label='车辆轨迹', color='b')
plt.plot(x_front_in_traj, y_front_in_traj, label='前内车轮轨迹
',color='y')
plt.plot(x_front_out_traj, y_front_out_traj, label='前外车轮轨迹
',color='r')
plt.plot(x_rear_in_traj, y_rear_in_traj, label='后内车轮轨迹
',color='g')
plt.plot(x_rear_out_traj, y_rear_out_traj, label='后外车轮轨迹
```

```
',color='maroon')
plt.scatter(x_traj[0], y_traj[0], color='r', marker='o', label='起
点')
plt.scatter(x_traj[-1], y_traj[-1], color='g', marker='o', label='
终点')
plt.axis('equal')
plt.xlabel('X (m)')
plt.ylabel('Y (m)')
plt.title('前轮驱动单车运动模型仿真')
plt.legend()
plt.show()



# 构建 DataFrame
columns = ['时间/s',
           '车辆中心_x', '车辆中心_y',
           '前内轮中心_x', '前内轮中心_y',
           '前外轮中心_x', '前外轮中心_y',
           '后内轮中心_x', '后内轮中心_y',
           '后外轮中心_x', '后外轮中心_y']
df_positions = pd.DataFrame(data, columns=columns)

# 打印 DataFrame
print(df_positions)

# 保存到 Excel 文件
df_positions.to_excel('result2.xlsx', index=False)
```

5. 出库问题

```
Python
import math
import numpy as np

# 计算内后轮

r_in = 0.2

L = 2.4

R = r_in + 0.92
```

```
psi = np.rad2deg(math.atan(L / R))

if psi >= 40:
    psi = 40

print('最大转弯角度为：', psi)
# 计算外前轮

r_out = 5.5 + 5.3 - 0.65 - 0.5 - 0.6 / 2

R = math.sqrt(r_out * r_out - L * L) - 0.92

psi = np.rad2deg(math.atan(L / R))

print('最小转弯角度为：', psi)
```

6. 蒙特卡罗法检验出库问题

```
Python
import random
import math
import numpy as np

# 车辆参数
vehicle_length = 4.0  # 车辆长度（m）
vehicle_width = 2.0  # 车辆宽度（m）
wheel_diameter = 0.6  # 车轮直径（m）
wheel_width = 0.16  # 车轮宽度（m）
front_rear_distance = 0.5  # 前后轮到车头车尾的距离（m）
L = 2.4

# 车位参数
parking_space_length = 5.3  # 停车位长度（m）
parking_space_width = 2.4  # 停车位宽度（m）
road_width = 5.5  # 道路宽度（m）

# 蒙特卡罗模拟参数
num_samples = 1000000  # 采样次数

# 初始化最大和最小转弯角度
max_turn_angle = 0
```

```
min_turn_angle = 50

# 初始化最大和最小转弯开始位置
max_start_turn_position = parking_space_length
min_start_turn_position = 0.0

# 蒙特卡罗模拟
for _ in range(num_samples):
    # 随机生成一个车辆开始转弯的位置
    start_position = random.uniform(0, parking_space_length)

    # 随机车辆在转弯过程中的最大和最小转弯角度
    max_angle = random.uniform(0, 40)
    min_angle = random.uniform(0, 40)

    # 更新最大和最小转弯角度
    if max_angle > max_turn_angle:
        R = L / math.tan(np.deg2rad(max_angle))
        R_in = R - 0.92
        if R_in * R_in - (R_in - 0.2) * (R_in - 0.2) >= (3.85 -
start_position) * (3.85 - start_position):
            max_turn_angle = max(max_turn_angle, max_angle)

    if min_angle < min_turn_angle:
        R = L / math.tan(np.deg2rad(min_angle))
        R_out = math.sqrt((R + 0.92) * (R + 0.92) + 2.4 * 2.4)
        if start_position + R_out <= 5.5 + 3.85:
            min_turn_angle = min(min_turn_angle, min_angle)

# 使用蒙特卡罗统计开始转弯位置
for _ in range(num_samples):
    # 随机生成一个车辆开始转弯的位置
    start_position = random.uniform(0, parking_space_length)
    # 更新最大转弯开始位置
    R = L / math.tan(np.deg2rad(max_turn_angle))
    R_in = R - 0.92
    if R_in * R_in - (R_in - 0.2) * (R_in - 0.2) >= (3.85 -
start_position) * (3.85 - start_position):
        max_start_turn_position = min(max_start_turn_position,
start_position)

# 再次使用蒙特卡罗统计开始转弯位置
for _ in range(num_samples):
```

```
    # 随机生成一个车辆开始转弯的位置
    start_position = random.uniform(0, parking_space_length)
    # 更新最小转弯开始位置
    R = L / math.tan(np.deg2rad(min_turn_angle))
    R_out = math.sqrt((R + 0.92) * (R + 0.92) + 2.4 * 2.4)
    if start_position + R_out <= 5.5 + 3.85:
        min_start_turn_position = max(min_start_turn_position,
start_position)

# 打印结果
print("前内轮最大转弯角度（角度）：", max_turn_angle)
print("前内轮最小转弯角度（角度）：", min_turn_angle)
print("最大转弯开始位置距离出库线（m）：", 3.85 -
max_start_turn_position)
print("最小转弯开始位置距离起点（m）：", min_start_turn_position)
```