清 华 大 学

综 合 论 文 训 练

题目：基于数据驱动的移动目标预测与拦截

系　　别：自动化系

专　　业：自动化

姓　　名：屈晨迪

指导教师：莫一林 副教授

2021 年 5 月 27 日

关于学位论文使用授权的说明

本人完全了解清华大学有关保留、使用学位论文的规定，即：学校有权保留学位论文的复印件，允许该论文被查阅和借阅；学校可以公布该论文的全部或部分内容，可以采用影印、缩印或其他复制手段保存该论文。

(涉密的学位论文在解密后应遵守此规定)

签　名：庞晨迪　导师签名：莫一林　日　期：2021.6.3.

中文摘要

近年来随着感知、视觉、控制规划等方面技术的逐步成熟，移动智能体在各个行业的应用愈来愈广泛。而如何控制智能体对环境中的移动目标进行拦截一直以来都是机器人技术中的一个重要问题，其研究成果可以应用于军事、商业、日常生活等多个领域。但在一些情况下，由于移动目标的速度和运动规律未知，且智能体可能遇到较为复杂的障碍，此时如何使智能体在较短时间内完成对目标的拦截，是一个尚未充分解决的难题。

针对环境中存在静态或动态障碍物时，具有不定运动规律的移动目标的拦截问题，本文设计了一套完整的算法流程。算法首先对移动目标进行观测，基于观测得到的历史轨迹数据，通过多项式拟合的方法在线预测目标的未来轨迹，接着利用目标函数辅助选取拦截点，最后采用路径和速度解耦的方式进行轨迹规划。算法充分考虑了车辆行驶过程中的可行性、安全性，满足车辆的运动学和动力学特征。同时自主搭建了仿真环境，在仿真小车上部署并测试了本算法，成功实现了移动目标在匀速直线、曲线等运动模式下的拦截。相较于诸多已有的拦截算法，本文提出的算法适用场景更广，可以满足大多数目标拦截任务的需求。

关键词：移动智能体；目标拦截；轨迹预测；轨迹规划

ABSTRACT

In recent years, as technologies such as perception, vision, and planning gradually mature, mobile agents have been applied to various fields. It has long been a critical topic of robotics to intercept mobile targets in the environment by controlling agents. The research can be translated into military, commercial or day-to-day applications. However, given that the speed and the motion of the moving target may not be known under some circumstances, and the agent might encounter rather complex obstacles, it remains a challenge to enable the agent of completing the interception of the target in a short period of time.

This paper designed a complete algorithm flow for solving the problem of intercepting moving targets with indeterminate motion laws where static or dynamic obstacles exist in the environment. The algorithm first observed the moving target, yielding the trajectory data which was used to predict the future trajectory of the target on-line via the polynomial fitting method. The algorithm then used an objective function to assist in selecting the interception point and finally planned a trajectory with path and velocity decoupling. The algorithm satisfied the kinematic and dynamic characteristics of the vehicle, with thorough considerations of feasible and safety during driving. Meanwhile, a simulation environment was built where the algorithm was deployed and tested on a simulation car, which successfully achieved the interception of moving targets under uniform straight, curved, and other motion modes. The algorithm proposed in this paper, compared with existing interception algorithms, could be used in a broader range of occasions for accomplishing most of the target interception tasks.

Keywords： Mobile Agent; Target Interception; Trajectory Prediction; Trajectory Planning

目　　录

第1章 引言

1.1 研究背景

一直以来，机器人都是全球主要发达国家关注的战略重点。从 2012 年开始，多个发达国家先后在国家层面就机器人领域推出了支持策略，希望能够抢占机器人技术发展的主动权。韩国于 2012 年发布战略书《机器人未来战略 2022》，期望本国在该领域能够跻身世界前列；2013 年美国发布了《机器人发展路线图》，其中明确列出了九个需要重点关注和研究的机器人领域；同年，德国提出了《工业 4.0 战略》，指出让机器人替代人力劳动，接管企业工厂的发展方向；而英国在 2014 年也发布了《机器人和自主系统战略 2020》，希望能在全球机器人市场占到 10%的份额[1]。

与上述国家相同，我国也对该领域十分关注，从中央到地方出台了多个指导机器人产业发展的相关政策。2017 年，国家多个部门联合发布了《国家机器人标准体系建设指南》，统筹安排我国机器人标准化建设，大力推动智能制造业的发展，下半年又推出了《中国机器人标准化白皮书》，深入分析了我国在机器人领域的需求，梳理了我国现有的相关技术和标准化工作，对我国机器人领域自主创业创新、产业健康发展和未来战略规划提供了有效的参考和支撑[1]。

伴随着国家层面的支持，加上近年来感知[2]、视觉[3]、控制规划[4]等方面技术的逐步成熟，移动智能体（包括自动驾驶汽车、无人机、AGV 等）在各领域的应用愈来愈广泛。于其相对人类能够更好地适应恶劣的环境条件，且具有更为优秀的隐蔽性、持续作业能力等，在监察、搜寻、目标攻击与拦截等军事领域均有广阔的前景。而移动机器人技术中一个尚未充分解决重要的问题，即使用机器人对目标进行跟随或拦截，其研究成果不仅可以应用在自动制导系统[5]、智能体编队破坏或攻击、自主监视等军事领域，也可以用于足球机器人技术[6]、家用生活机器人等商业领域。同时，智能体在运动过程中，还需要与周边的动态环境进行实时交互，对一段时间内的环境变化进行预测并做出相应的控制决策和规划，这也是当前多个方向如自动驾驶的热门研究要点。另外，伴随人工智能浪潮的来临，诸多机器学习算法的提出为该问题研究注入了新的活力，使用神经网络识别

或习得移动目标的运动规律，用模仿学习[7]或强化学习[8]帮助智能体进行拦截，从而摆脱控制相关的基础算法，也是当前许多团队研究的方向。

1.2 国内外研究现状

近年来，学界对各种情景下移动目标拦截的问题有多方面的研究。其中一个热门应用场景是自动制导的导弹拦截、反拦截系统，如多发导弹协同作用拦截高超声速目标，或拦截低空飞行的慢速小目标导弹。田宪科等人[9]利用仿真软件研究了导弹拦截的几点影响因素，建立了导弹拦截点的计算模型；杨子成等人[10]采用监督学习的方法，设计了神经网络计算拦截时间和拦截点。但导弹的拦截研究不能扩展应用到本项目的问题上，由于导弹本身具有高速运动的特点，其拦截点附近的运动可近似为匀速直线运动。而本文研究的移动目标轨迹不定、速度未知，且环境中存在障碍物，因此两种场景间不能做算法的迁移。

在本文场景下，即使用移动机器人进行目标拦截，如何使机器人沿路程、时间较短的路径拦截目标是一个尚未充分解决的难题，F.Belkhouche 等人[11]研究了有障碍物存在时轮式机器人对运动目标的跟踪和拦截，他们基于几何规则和运动学方程提出了一种指导策略，并根据该策略设计了一个闭环控制率，其中机器人的角速度等于视线角的转动速率；Yizhaq Meyer 等人[12]在规划智能车拦截路径时采用了 dubins 规划，这里的 dubins 规划指找到一条连接起点和目标点的 dubins 曲线[13, 14]，它是在满足了曲率要求和始末速度切线方向的条件下，连接平面上两点的最短路径，因此常被用于车辆的路径规划。Yizhaq Meyer 等人的文章证明了在某些条件下 dubins 路径即为最短拦截路径，同时提出了三种路径延申的算法，保证了拦截的成功率，但该实验并未考虑障碍物存在的情况；Ian R.Manchester 等人[15]研究了基于视觉的拦截算法，该策略没有路径规划和跟随两个阶段，只使用摄像机提供的拍摄数据，即可以实现从特定角度拦截目标；Qingbao Zhu 等人[16]研究了当空间中存在复杂障碍物时，使用蚁群算法快速规划到拦截点的导航路径，机器人沿规划出的短直线组合运动，并在拦截过程中不断观测、预测、更新路径，直至完成拦截，但该方法规划出的路径不平滑，不满足车辆运动学特性。

在一般拦截问题的研究中，系统首先需要对移动目标的未来轨迹做出预测，之后需规划出符合预期的拦截路径，考虑到单独讨论轨迹预测的文献较少，因此

这里将调研范围扩大至目标跟随或追踪任务，可以给轨迹预测提供指导思路，下面分别进行讨论。

1.2.1 目标跟踪文献调研

移动机器人在目标跟踪方面已经取得了许多成果。基于模糊控制的跟踪是常见的方法之一，[17, 18]中 R. Luo 等人提出了一种全新的自适应运动控制算法，结合了基于灰色理论的位置预测方法和用于移动目标跟踪的超前模糊控制，但模糊控制方法有一些明显的缺陷，如计算效率不高、具有不确定性等，有时难以获得最优解。Jing Chen 等人[19]主要探讨了无人机在复杂环境中对移动目标的在线跟随及避障问题，文章的主要贡献是提出了一种路径规划算法，可以在保证安全避障（障碍物约束）和动力学可行性（动力学约束）的条件下，二次规划（QP）生成一条平滑路径，实现目标的追踪。该研究中通过多项式拟合预测未来轨迹，并加入加速度约束项防止过拟合的方法可以应用于本次课题。在 Ian R.Manchester 等人[15]基于视觉拦截的方法中，结合拦截机器人的运动状态及观测到目标与自身的位置差，使用扩展卡尔曼滤波（EKF）的方法对移动目标的运动状态进行估计和预测，卡尔曼滤波是一个有五十多年历史的模型预测算法，到目前为止在很多领域都有应用并起到了很好的效果，但基础的卡尔曼滤波算法要求对观测系统的内部状态和运动模型已知，而本课题中运动目标的仅有位置状态是可观的，因此难以应用。此外，在自动驾驶领域还有用神经网络模型 LSTM 进行预测的方法，如 Florent 等人[20]利用 LSTM 预测车辆在高速路上的驾驶轨迹，但深度学习相关的方法往往需要较为庞大的训练数据作为前提，无法保证训练的效率和最终能否收敛，不适用于本课题需要实时预测的要求。

1.2.2 路径规划文献调研

在路径规划控制领域，基于优化的控制是一种传统经典的方法，但优化算法通常需要较大的计算量，且设计函数凸性的问题，在场景中有障碍物的情况下可能要做凸近似，进一步增大了计算量，会使路径规划过程非常耗时，影响拦截的实时性能，如 Mark W.Mueller 等人用最优控制的思想解决了四旋翼无人机的路径规划问题[21]。近年来各种势场方法也被广泛应用于机器人路径规划领域，[22]中作者提出了一种全新的势场方法，用于存在动态障碍物情况下的路径规划，[23]中作者使用势场方法进行机器人的速度规划。但势场法在静态环境的规划下也存在一些明显的缺点，如容易陷入局部极小值，并可能发生震荡和死锁的问题，这

些缺点在目标和障碍物动态运动起来后会更加严重。[11]基于几何规则的策略也是一种较为有效的方法，机器人可以使用这种方法沿短路径运动至目标处，但如果环境中存在更加复杂的障碍，该方法的实施会变得很困难。此外，基于视觉的方法如[15]也可以实现目标的追踪或拦截，但视觉方法存在两个问题，一是这些算法需要处理大量的图片数据，难以保证拦截的实时性，二是需要在运动过程中始终保持移动目标必须在实验用的摄像机拍摄范围内，这需要摄像机的实时校准，否则其精准性就难以保证。在[24]中，作者提出了一种有效的基于网格的距离传播动态系统，用于动态环境中实时的机器人路径规划，该系统设置了局部的惩罚函数和障碍物周围的安全裕度，通过最小化当前位置到目标点的已知距离与沿路径累计的惩罚之和，规划出机器人的行驶路径，该算法类似于 D*搜索，但不需要维护待更新点的队列。

上述方法均取得了较为良好的效果，但仍在不同程度上存在未解决或未考虑到的问题，有的算法难以满足实时性要求，有的算法没有考虑车辆的运动学、动力学约束，有的没有考虑障碍物存在的情况等，因此在该领域还需要持续性的研究。

1.3 选题及主要工作

本文的选题为基于数据驱动的移动目标预测与拦截，为了简化研究问题，移动目标和观测跟随的智能体均为各方面配置相同的智能小车，实验中移动目标以一定规律在一个置有障碍物的连通场景中运动，初步设定其运动模式为匀速直线运动、匀变速直线运动、匀速曲线运动等基础运动模式中的一种。题目基于数据驱动的意义在于，系统对移动目标的运动模型没有先验知识，仅有在线观测到的位置坐标作为输入。本文针对该拦截问题设计了一套完整的算法流程，系统根据历史观测数据实时预测移动目标在未来有限时刻内的运动轨迹，在该轨迹上确定最优的拦截点位，继而通过一些轨迹规划算法规划出一条最优的路径及路径上的速度、加速度信息，在避开障碍物的同时对目标进行有效的拦截。

1.4 论文结构与主要贡献

本文的其他部分安排如下：

第 2 章，详细介绍了本次课题的总体技术路线，该项目可以具体分为四个模块，分别是感知、预测、决策与规划和控制模块，其中感知模块为传感器检测发现目标，观测其运动状态，获取目标位置和拦截智能车的位置；预测模块根据观测得到的历史位置数据，预测目标的未来运动轨迹；在决策与规划模块中选择合适的拦截点，进行轨迹规划；再由底层控制模块控制车身跟随航迹点。感知、预测和控制模块会在第 2 章做一讨论，而决策规划模块将会在第 3 章详细展开。

第 3 章，详细探讨决策与规划模块的设计，包括方案选择和最终确定的算法流程，本章解释了如何在拦截问题中融入分层优化的思想，介绍了拦截点选取的方法和解耦的轨迹规划算法，并对具体算法的时间效率加以探讨。

第 4 章，介绍了本次主要用到的实验平台 ROS 与仿真环境 Gazebo，说明了仿真环境中的搭建细节和过程。

第 5-6 章，总结本论文的研究成果和结论，对实验结果进行了分析，指出了实验中仍旧存在的几处不足，并提出了对未来可能的研究方向。

本文的主要贡献在于，在环境中存在障碍物的情况下，提出了一种移动目标的拦截方法，该策略通过拦截点的选取和拦截轨迹规划，在完成任务的同时考虑了较短的拦截时间和较少的能量消耗，且规划出的路径满足可行性和车辆动力学特性。本文设计的算法可以应用于具有不定轨迹的移动目标在包含静态、动态障碍物环境下的拦截问题，相较于已有的诸多拦截算法适用范围更广、更一般化。

第2章 总体技术路线

2.1 概述

本章将详细讨论课题的各部分技术路线。智能小车对移动目标进行预测和拦截的任务，可以考虑成一个控制问题，传统的控制系统流程如图 2.1 所示，外界输入数据到控制器，经过一些内部处理和运算后输出结果，再通过传感器对系统输出进行检测，作为反馈信息一并输入到控制器，形成闭环反馈的稳定控制。

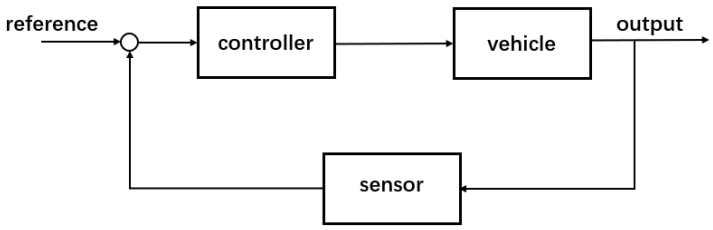


图 2. 1 传统控制系统

智能车系统保留了传统控制系统的主要流程，并另外加入了一些新模块。如图 2.2 所示，除了原有的控制和感知模块外，新增了预测和决策规划模块。传感器得到的数据不仅作为反馈输入，还可以用于系统对目标的定位和检测（追踪），如摄像头获取图片信息，或激光测距仪获得距离信息，都可以确定目标所在的位置。在感知定位完成后，将位置信息输入到预测模块，对目标接下来的运动状态做预测和估计，再利用预测的结果帮助系统进行行为决策和运动规划，运算完成后，将决策规划信息交付控制器，再由底层控制器最终控制智能车的运动。

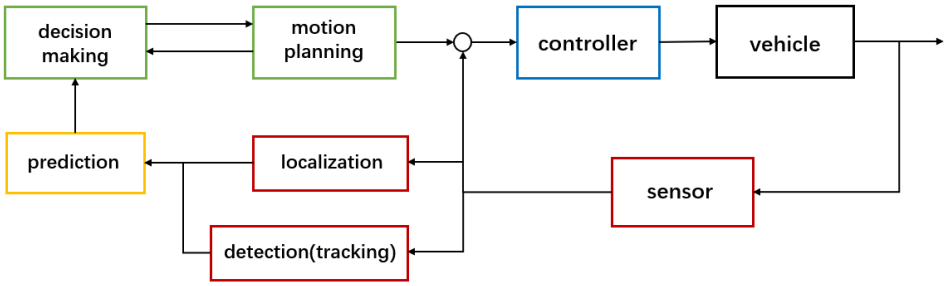


图 2. 2 智能车控制系统

根据上述智能车规划任务的基本流程，由此，本次课题也可以按照上述流程从总体上分为四大模块，分别是感知、预测、决策规划和控制模块。在感知模块中，首先系统内部传感器检测发现目标，在一个时间段$[t_l, t_0]$观测记录目标的位置信息，保存为历史数据，同时感知模块还需要在后期控制拦截小车运动的时候实时返回小车的位置。感知模块将目标的历史运动数据交给预测模块，由预测模块进行未来一个时间段$[t_0, t_p]$的轨迹预测，之后决策和规划模块会在预测出的轨迹上按照一定要求选取最优的点位作为拦截点，并生成一条满足可行性和安全性的路径，最后由控制模块控制拦截小车遵循规划路径运动至拦截点处，完成一次拦截任务。整体流程如图 2.3 所示，其中最为核心的是决策规划部分，将在第 3 章中做重点介绍。

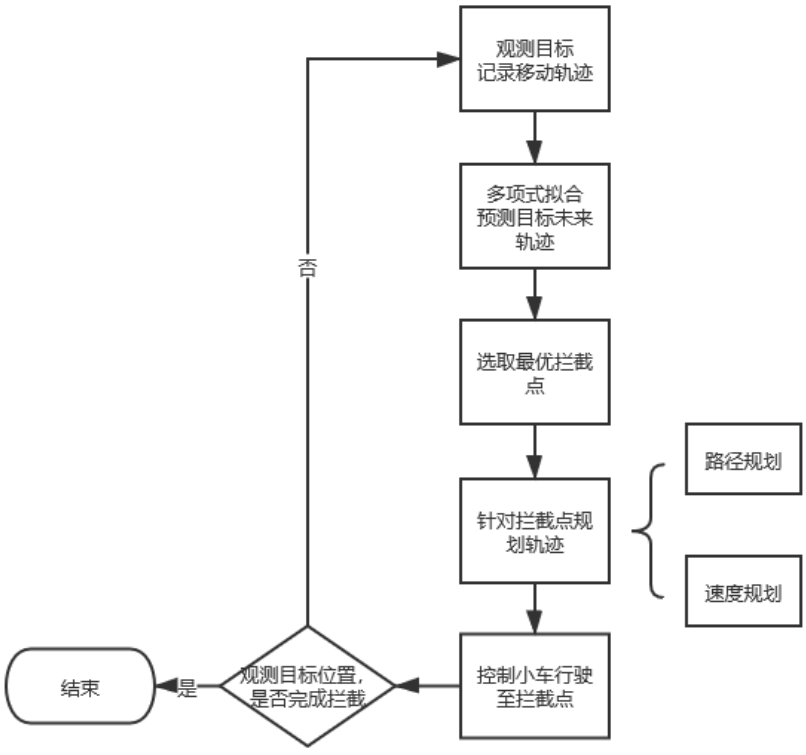


图 2. 3 拦截任务流程图

下文将会详细讲解各模块的方案选择和最终设计思路。

2.2 感知模块

2.2.1 实现方案概述

在本项目中，感知模块需要检测发现目标，并能够实时获取目标和拦截小车自身的位置信息。感知模块在发现移动目标后，观测、记录一段时间内目标的移动轨迹，作为历史位置储存在系统中，并提供给之后预测模块进行基于数据的轨迹预测。

在 ROS + Gazebo 搭建的仿真环境①中，可以通过订阅服务消息“/gazebo/get_model_state”来实现。向 Gazebo 发送了名为“get_model_state”的请求后，收到的返回消息包括仿真机器人的 Pose 和 Twist 信息，其中 Pose 包含三维空间位置 position (x, y, z)和四元数形式的姿态信息 orientation (x, y, z, w)，而 Twist 包含三个方向的线速度 linear (x, y, z)和三个方向的角速度 angular (x, y, z)，如图 2.4 所示。通过该服务话题可以很方便地在仿真中获取所需要的位置信息。

```
qucd@qucd-Lenovo-Legion-Y7000-1060:~$ rossrv show gazebo_msgs/GetModelStatestring model_name
string relative_entity_name
---
std_msgs/Header header
  uint32 seq
  time stamp
  string frame_id
geometry_msgs/Pose pose
  geometry_msgs/Point position
    float64 x
    float64 y
    float64 z
  geometry_msgs/Quaternion orientation
    float64 x
    float64 y
    float64 z
    float64 w
geometry_msgs/Twist twist
  geometry_msgs/Vector3 linear
    float64 x
    float64 y
    float64 z
  geometry_msgs/Vector3 angular
    float64 x
    float64 y
    float64 z
bool success
string status_message
```

图 2. 4 get_model_state 服务消息格式

而若需要在实际场地中进行实验，则需要在实验场地上方搭建运动捕捉摄像头系统，如图 2.5 所示，该系统 FPS 约为 240Hz，对智能小车的定位可以精确到毫米级。当然，也可以选择在拦截小车上搭载双目摄像头和激光雷达，从而获取周围的环境信息和目标距自身的位置，同样可以进行后续的任务流程，这样可以减少对设备的依赖，提升系统实用性，但精度会受到一定影响。

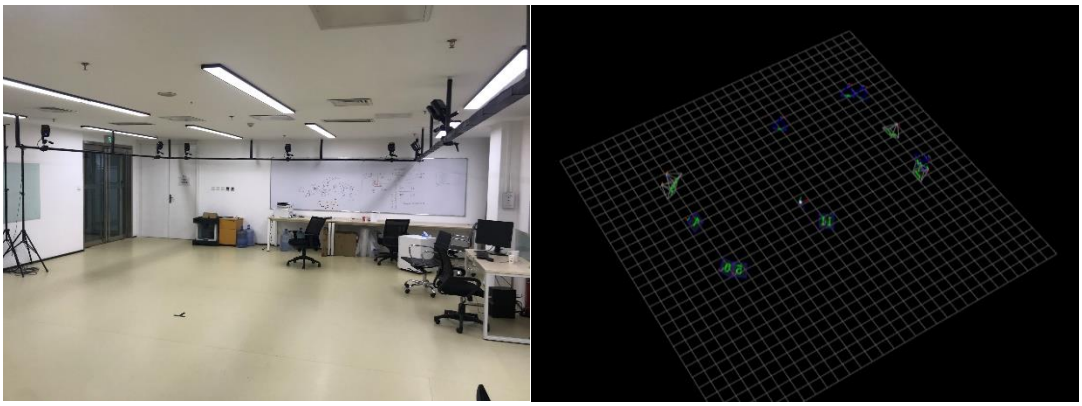


图 2. 5 真实场地搭建

2.2.2 卡尔曼滤波提高观测精度

在仿真环境下通过订阅服务话题得到的位置信息精度都是很高的，但实际情况下位置的测量可能存在噪声干扰，导致结果有误差，如何减小误差是值得讨论的问题。

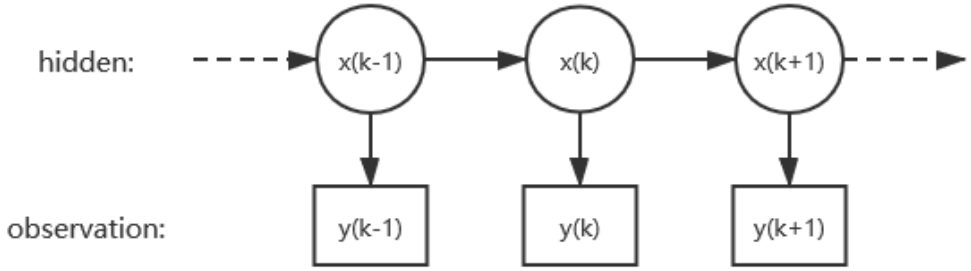


图 2. 6 隐马尔科夫模型

在轨迹预测的文献调研部分，提到过卡尔曼滤波的方法。卡尔曼滤波[27]是一种效率较高的递归滤波器，当观测和预测都存在满足高斯分布的误差时，利用此种滤波方法能获得系统状态的最优估计。从数学本质上讲，卡尔曼滤波是图 2.6 所示模型应用贝叶斯方程得到的显示解。因为实验中移动目标的运动模型，即状态转移矩阵未知，虽然可以假设一个转移矩阵，但精度会达不到要求，因此无法应用该方法做轨迹预测，但卡尔曼滤波将当前观测数据和历史记录数据融合起来的特点可以用于提高位置测量的精度。在本课题讨论的问题背景下，可以用一个四元数组(x_1, x_2, x_3, x_4)来对移动目标的状态进行刻画，其中(x_1, x_2)是目标的位置

坐标，(x_3, x_4)是目标在两个方向上运动的分速度，则移动目标的运动和观测模型可以表示如下：

$$\begin{aligned} x_k &= A\,x_{k-1} + q_{k-1} \qquad q_{k-1} \sim N(0, Q) \\ y_k &= H\,x_k + r_k \qquad r_k \sim N(0, R) \end{aligned} \tag{2-1}$$

卡尔曼滤波算法分为预测和更新两部分，对系统状态量x_k和其协方差P_k进行迭代，具体步骤如下：

预测：

$$\begin{aligned} x_k^- &= A_{k-1} x_{k-1} \\ P_k^- &= A_{k-1} P_{k-1} A_{k-1}^T + Q_{k-1} \end{aligned} \tag{2-2}$$

更新：

$$\begin{aligned} K_k &= P_k^- H_k^T (H_k P_k^- H_k^T + R_k) \\ x_k &= x_k^- + K_k (z_k - H_K x_k^-) \\ P_k &= P_k^- - K_k (H_k P_k^- H_k^T + R_k) K_k^T \end{aligned} \tag{2-3}$$

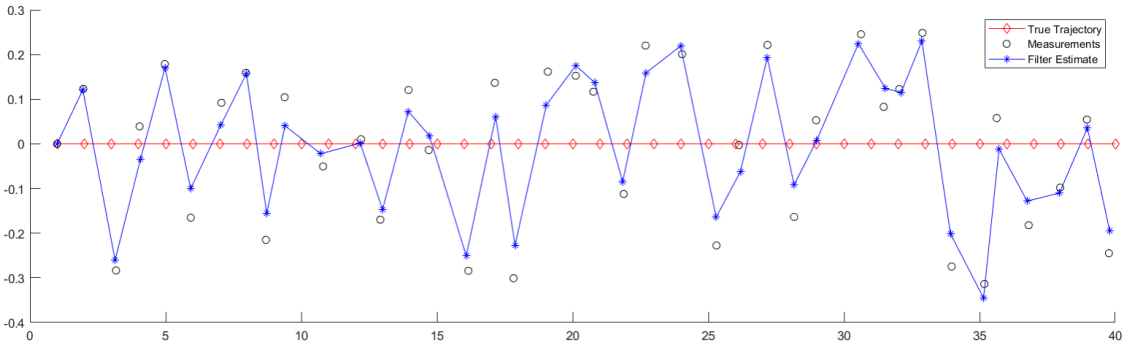
其中 k 为迭代轮数，x_k表示要估计的系统状态，z_k为传感器的观测数据，A_k为系统状态的转移矩阵，Q 和 R 分别是运动和观测模型误差的协方差矩阵，代表了模型预测值和测量值的权重，当 R 不变时，Q 越大，则估计越信任测量值，Q 无穷表示只使用测量值，反之 Q 越小越偏向预测值，Q 为零则代表只进行模型预测；K_k为迭代过程的中间变量，与 Q 和 R 的比值有关。

本实验中，A、H、Q、R 的参数选择如式（2-4）所示，其中q_1^c, q_2^c分别是过程噪声在两个方向上的连续时间方差，σ_1^2, σ_2^2是 x 和 y 坐标对应的观测噪声方差。在 MATLAB 中测试了目标匀速运动和匀加速运动两种情况下的滤波效果，设置$q_1^c = q_2^c = 0.2$，$\Delta t = 1$，运行$N = 40$步，并求出滤波前、滤波后的误差平方和，计算误差减小的百分比。

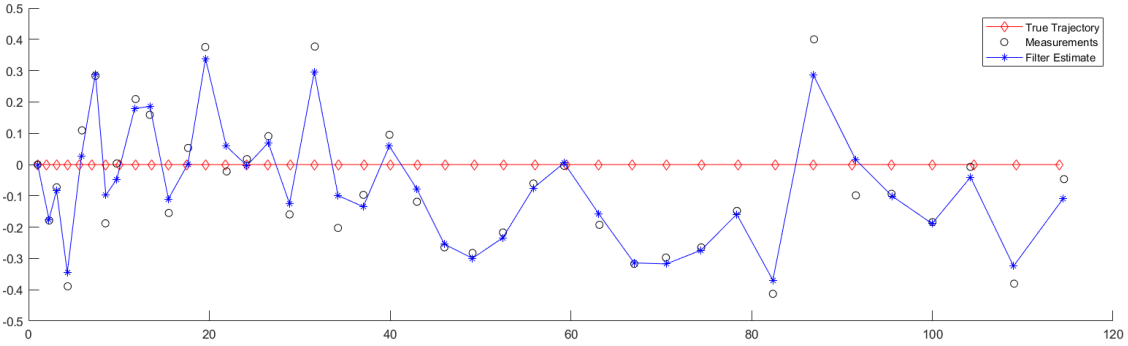
$$A=\begin{pmatrix}1&0&\Delta t&0\\0&1&0&\Delta t\\0&0&1&0\\0&0&0&1\end{pmatrix},\quad Q=\begin{pmatrix}\frac{q_1^c\Delta t^3}{3}&0&\frac{q_1^c\Delta t^2}{2}&0\\0&\frac{q_2^c\Delta t^3}{3}&0&\frac{q_2^c\Delta t^2}{2}\\\frac{q_1^c\Delta t^2}{2}&0&q_1^c\Delta t&0\\0&\frac{q_2^c\Delta t^2}{2}&0&q_2^c\Delta t\end{pmatrix}$$

$$H=\begin{pmatrix}1&0&0&0\\0&1&0&0\end{pmatrix},\quad R=\begin{pmatrix}\sigma_1^2&0\\0&\sigma_2^2\end{pmatrix}\tag{2-4}$$

当$\sigma_1=\sigma_2=0.2$时，移动目标在两种运动模式下的卡尔曼滤波效果如图 2.7 所示，图中红色菱形表示目标的真实轨迹，黑色圆圈为加了噪声后的传感器测量信息，蓝色星形为滤波后结果，可以看到滤波后总体误差有较明显的减小。



(a) 1m/s 匀速运动



(b) $0.1\mathrm{m/s^2}$匀加速运动

图 2. 7 两种运动情况下滤波效果

计算在不同$\sigma_1 = \sigma_2$下的误差平方和，如表 2.1 所示，可以看到在匀速和匀加速两种模式下使用卡尔曼滤波，与真实轨迹的误差在滤波前后均有减小，减小程度基本持平，匀加速时误差较大，误差减小百分比也较大。同时，可以看到 Q 不变时，随着观测误差协方差矩阵 R 的增大（也即σ值增大），误差逐渐增大，滤波效果也逐渐增强，这符合之前关于 Q 和 R 比值的讨论，即 Q/R 越大，滤波越依赖观测数据。实际情况中，需要在实验中不断调整 QR 比，以找到合适的σ值。总而言之，当感知模块受到噪声干扰时，卡尔曼滤波有效地提高了观测精度。

表 2. 1 不同σ(R)下误差平方和比较

	误差平方和	$\sigma = 0.05$	$\sigma = 0.2$	$\sigma = 0.5$
匀速运动	滤波前	0.1875	3.0839	15.4623
	滤波后	0.1816	2.5693	10.7318
	误差减小	3.15%	16.69%	30.59%
匀加速运动	滤波前	0.2152	3.5232	19.8420
	滤波后	0.2072	2.8633	11.3466
	误差减小	3.72%	18.73%	42.82%

2.3 轨迹预测模块

2.3.1 多项式拟合预测

前文介绍了系统的感知模块，得到了移动目标的历史位置数据后，需要在预测模块预测目标的未来轨迹。考虑到在一般情况下，移动目标的运动轨迹不会太复杂，而多项式回归计算效率高，鲁棒性好，因此使用多项式回归的方法预测未来轨迹是可行的。多项式回归是线性回归模型的一种，有 y 关于 x 的多项式，给定最高次数n，通过一系列已知数据(x_i, y_i)拟合出多项式 n+1 个系数a_0~a_n。

在本次实验中，由感知模块观测记录了一系列历史位置和时间数据，用三元数组存储为(x_i, y_i, t_i)，其中(x_i, y_i)为t_i时刻目标所在的坐标。令移动目标t时刻所在位置$T(t)$为一个二维函数，有$T(t) \in R^2, t \in R$，对$T(t)$做如下所示的泰勒展开后，忽略次数大于n_t的高次项，则将$T(t)$转化为最高次项为n_t的多项式$\hat{\mathrm{T}}(\mathrm{t})$。

$$
\begin{aligned}
T(t) &= \sum_{i=0}^{\infty} \frac{T^{(i)}(0)}{i!} \times t^i \\
&= \sum_{i=0}^{n_t} \frac{T^{(i)}(0)}{i!} \times t^i + O(t^{n_t}) \\
&\approx \begin{bmatrix} \frac{T^{(0)}(0)}{0!} \\ \frac{T^{(1)}(0)}{1!} \\ \vdots \\ \frac{T^{(n_t)}(0)}{n_t!} \end{bmatrix}^T \begin{bmatrix} 1 \\ t \\ \vdots \\ t^{n_t} \end{bmatrix} \\
&= \hat{T}(t)
\end{aligned} \tag{2-5}
$$

得到待拟合多项式后，需要确定最小化的目标函数。将t_i时刻观测到目标的历史位置数据记为$p_i \in R^2$，p_i是一系列二维坐标点位，令t_0为当前时刻，在$[t_l, t_0]$时间段内观测记录了 L 个数据，需要最小化的误差项可以表示为：

$$
\min_{\hat{T}(\cdot)} \sum_{i=0}^{L} \| \hat{T}(t_i) - p_i \|_2^2 \tag{2-6}
$$

我们可以利用最小二乘法解决该优化问题，最小二乘法通过最小化误差的平方和，即公式 2-6，来求解已知数据的最优函数匹配，这里用解线性方程组的方法。对于每一组(x, y, t)，设置列向量$T[n]$，有$T[0] = 1$，$T[i] = T[i-1] * t$，矩阵 A 为 L 组数据$T_n * T_n$的和，A 为$n \times n$方形矩阵，b 为 L 组数据$T_n * (x, y)$的和，b 为$n \times 2$形状的矩阵，所求多项式参数记为 x，x 为$n \times 2$矩阵，则有：

$$
\begin{aligned}
(A^T A)x &= A^T b \\
x &= (A^T A)^{-1} A^T b
\end{aligned} \tag{2-7}
$$

对于式 2-7，由于待求解方程组的系数矩阵是对称矩阵，可以使用 LDLT 分解法进行求解，计算量较小。

当待拟合多项式次数较高时，多项式回归容易出现过拟合现象，通常而言，模型误差包括偏差、方差和一部分不可消除的误差。若拟合模型本身出现问题，如用线性函数去拟合非线性相关的数据，则会导致偏差较大，称作欠拟合；若使

用的拟合模型太过复杂，如本实例中设置的多项式次数n_t过大，则会出现方差较大的情况，称作过拟合。在多项式回归的过程中，应尽量避免出现欠拟合或过拟合，本实验不会出现欠拟合情况，针对过拟合问题，可以考虑加入正则化项，一般的正则化项有岭回归或 LASSO 回归，分别对应了 L2 范数和 L1 范数。本场景中我们有先验知识，即移动目标的位置在观测的时间段内不会发生太大变化，因此可以以此为基础设计正则化项，目标函数由式 2-6 变为式 2-8，其中$[t_l, t_0]$是观测记录移动目标的时间段，$[t_0, t_m]$是系统预测计算未来轨迹消耗的极短的一段时间，λ_t是正则化项的比例系数，可以调整约束程度。

$$\min_{\hat{T}(\cdot)} \sum_{i=0}^{L} \| \hat{T}(t_i) - p_i \|_2^2 + L\lambda_t \int_{t_l}^{t_m} \| \hat{T}^2(t) \|_2^2 \, dt \tag{2-8}$$

此外，轨迹预测也可以采用 LSTM 等机器学习的方法，但此类方法需要前期输入大量数据进行训练，消耗较大的计算资源和时间，才能达到期望的精度，且模型泛化性能不一定好，不适用于本实验中实时在线预测的要求，因此综合来看，多项式拟合简单、快速，是较好的选择。

2.4 底层控制模块

本节介绍底层控制模块，该模块的主要作用为控制小车循迹，首先对智能小车进行建模，之后介绍 PID 控制算法。

2.4.1 小车运动学模型

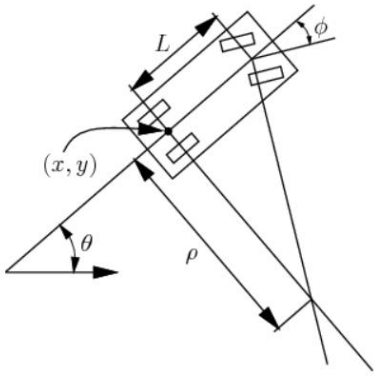


图 2. 8 小车运动学模型

要研究智能小车的控制算法，首先要对小车进行建模。这里使用一种常见的车辆运动学模型——自行车模型，如图 2.8 所示，将小车看作在平面上运动的刚体，车体总共有三个自由度，速度有两个自由度，其可行空间$C = R^2 \times S^1$，图 2.9 中表明了几个相关的运动参数，记车辆所处位置$q = (x, y, \theta)$，ϕ是小车的转向角，当ϕ为负数时代表朝规定的相反转向，前轮和后轮之间的距离记为 L，如果转向角ϕ固定，小车会做圆周运动，转弯半径为ρ，这里有$\rho = L/\tan\phi$. 定义了上述符号之后，小车的运动模型可以表示为：

$$\begin{aligned} \dot{x} &= f_1(x, y, \theta, s, \phi) \\ \dot{y} &= f_2(x, y, \theta, s, \phi) \\ \dot{\theta} &= f_3(x, y, \theta, s, \phi) \end{aligned} \tag{2-9}$$

在一个很短的时间Δt内，小车沿车轮朝向的方向行驶一小段距离，令$\Delta t \to 0$，有$dy/dx = \tan\theta$，又有$dy/dx = \dot{y}/\dot{x}$且$tan\theta = \sin\theta / \cos\theta$，则有约束：

$$-\dot{x}\sin\theta + \dot{y}\cos\theta = 0 \tag{2-10}$$

式 2-10 所示的约束在$\dot{x} = \cos\theta, \dot{y} = \sin\theta$时成立，且在该组解成比例扩大时，约束同样成立，这里的比例与车辆的线速度 s 直接相关，因此运动模型中前两个量可以是$\dot{x} = s\cos\theta, \dot{y} = s\sin\theta$.

接下来推导$\dot{\theta}$，令 w 是小车走过的距离（即速度的积分），若转向角固定，有$dw = \rho d\theta$，又$\rho = L/\tan\phi$，则有：

$$d\theta = \frac{\tan\phi}{L} dw$$

两边同时除以dt，考虑到$\dot{w} = s$，有

$$\dot{\theta} = \frac{s}{L}\tan\phi \tag{2-11}$$

至此，智能小车的运动模型建模完成，假设小车的线速度和角速度分别为$u = (u_s, u_\phi)$，$U = [-1, 1] \times (-\phi_{max}, \phi_{max})$，有

$$\begin{cases} \dot{x} = u_s \cos\theta \\ \dot{y} = u_s \cos\theta \\ \dot{\theta} = \frac{u_s}{L} \tan u_\phi \end{cases} \tag{2-12}$$

2.4.2 PID 控制算法

系统经过决策规划模块之后，得到了一系列路径点位和每段的速度取值，需要控制小车按照规划的速度循迹行驶至目标点，小车循迹的控制方法有多种，最常用的包括 PID 算法[28]和 MPC/迭代 LQR 算法。MPC 是模型预测控制，可以求得有等式或不等式约束情况下线性、非线性系统的最优控制，在智能小车上有较好的应用，可以保证车辆行驶的安全性、可行性和舒适性，但也存在参数难以调节、计算量较大等问题。综合考虑下本实验选用 PID 作为小车的底层控制算法。

PID 因为其简单而有效的特点，是当前业界常用的一种控制算法。PID 算法用到了负反馈促使系统趋于稳定的思想，流程框架图如图 2.9 所示，包含了比例、积分、微分三种运算，具体公式如下：

$$u = K_p \cdot e + K_i \cdot \int e dt + K_d \cdot \frac{de}{dt} \tag{2-13}$$

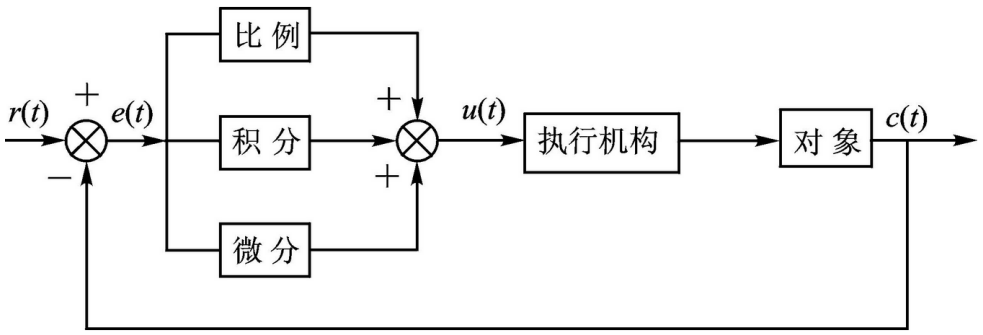
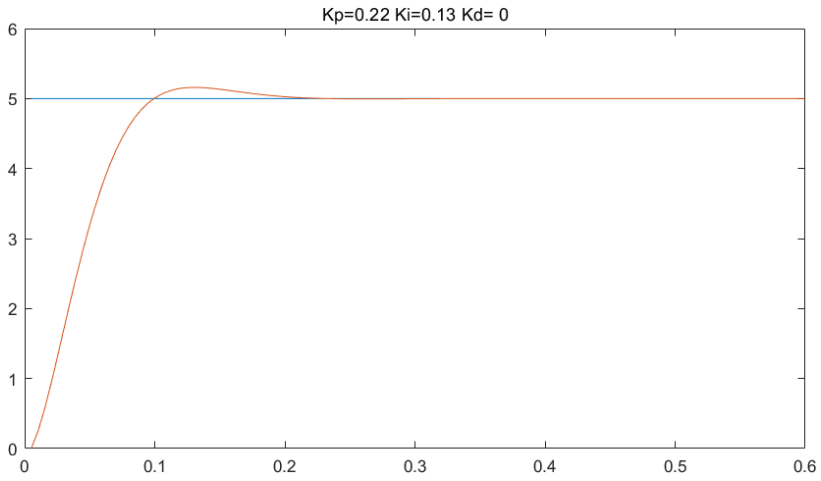


图 2. 9 PID 控制流程图

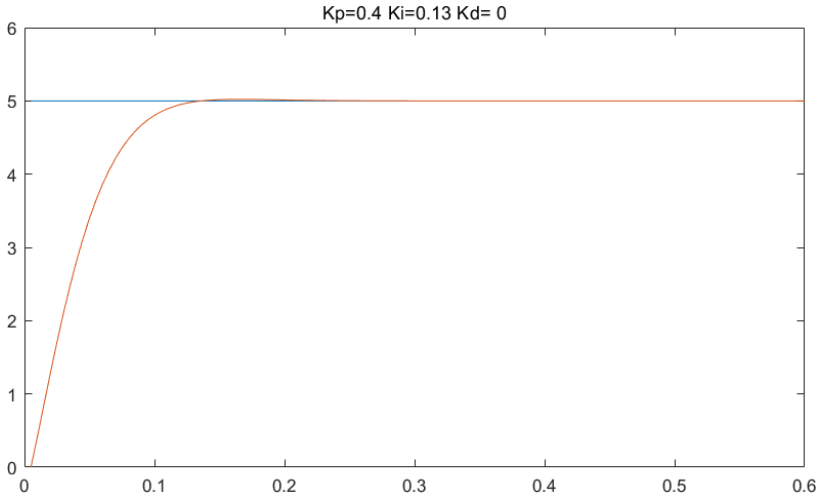
在智能小车的控制上应用 PID 控制，不需要知道小车内部的运动模型，只需得到系统偏差即可，包括位置偏差、速度偏差和航向角偏差。比例部分可以有效减少误差，使系统变量达到设定的大小，但比例系数设得太大，会出现超调等振荡现象，破坏小车行驶稳定性；积分可以用来减小静态误差，当小车收到外界环境干扰，如风向影响偏离航道，积分作用可以不断累积克服静态差，使小车保持

在正确的航向上；微分项可以减少系统的震荡，在小车有偏航趋势时及时更正，但微分项受扰动影响较大，在本例中可以仅用 P（比例）一项或 PI（比例积分）两项进行控制。在实际应用中，PID 的各项参数都需要在实验中不断调整，以找到一个最优的参数组合。

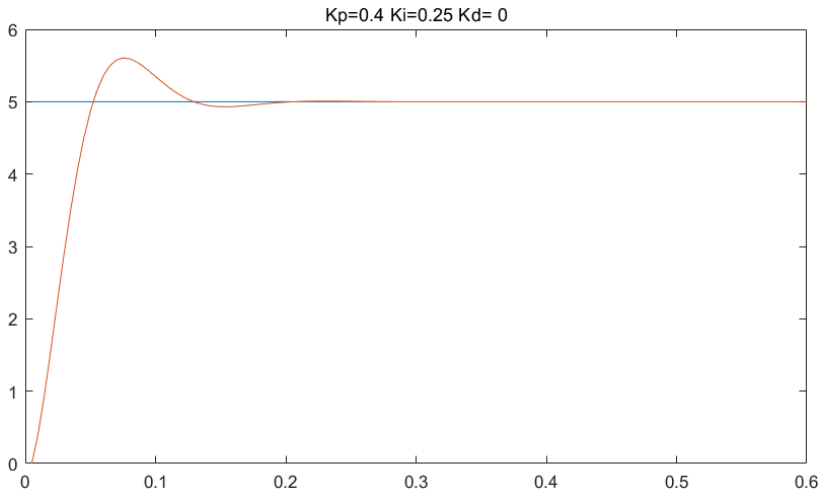
如图 2.10，是 PI 控制器在不同K_p和K_i参数下的仿真情况，可以看到，当积分系数不变，比例系数变大时，曲线有越来越明显的超调和震荡现象，而当增加积分系数后，超调现象有所减小，图 2.10(b)所示的控制是比较理想的情况。



(a)



(b)



(c)

图 2. 7 PI 参数调整效果

本章主要介绍了算法中的感知、预测和控制三个模块。其中感知模块采用卡尔曼滤波提高了噪声干扰下的观测精度，预测模块选用多项式拟合的方法拟合得到移动目标的未来轨迹，控制模块采用 PID 算法，实现了稳定的速度控制。

第3章 决策规划模块

3.1 概述

决策规划模块是本项目中最为重要的一部分，因此将其从第 2 章摘出另成一章。本章将详细讲述决策规划模块的整体思路、方案选择和最终实现方法。智能小车要对移动目标进行拦截，获得了目标未来一段时间$[t_0, t_p]$内的运动轨迹后，需要决策在轨迹上的哪个位置进行拦截，即选取哪个位置作为拦截点。其次，小车需要规划出一条符合时间要求和动力学、运动学约束的路径行驶到目标拦截点，完成拦截过程。因此，全程可以划分为两个步骤，最优拦截点的选取和最优轨迹的规划。

本模块采用分层优化的思想，首先通过目标函数辅助选出拦截点，再以该拦截点作为目标规划轨迹。当然，也可以采用一步优化的策略，直接规划出到拦截点集中每个点的最优轨迹，再选取最优的一条，但最优轨迹规划过程中要求解大量带约束的 QP 方程，这样一步式的策略会导致算法计算量增大，效率下降，不适用于要求实时在线拦截的系统；同时，直接对每个点轨迹规划可能会在目标点受到扰动时出现不稳定的情况，而分层优化会使系统更鲁棒。因此在综合考量下采用分层优化。

下面将分别阐述拦截点选取和轨迹规划。

3.2 选取最优拦截点

3.2.1 目标函数设计

要选取最优的拦截点，本质上是一个优化问题，要求解优化问题首先应设计一个好的优化目标，这里需要先明确什么是“最优”拦截点。本项目中有两点要求，一是拦截时间较短，二是拦截过程中小车能量消耗较少，最符合这两个要求的点位即理想拦截点。因此，按照上述“最优”的描述，设计目标函数如下：

$$\min_{D,T} \rho T + \int_0^T a^2 \, dt \tag{3-1}$$

其中 T 是所需要的拦截时间，ρ是比例系数，调节时间因素在优化目标中所占的比例，D 是一条到拦截点的路径，$J = \int_0^T a^2\, dt$是小车在路径 D 上的加速度平方和，反应了拦截过程中的能量消耗。

3.2.2 初步路径搜索

在预测出的未来轨迹上，等时间间隔Δt的选取一系列由二维坐标和时间组成的三维向量$P_i = (x_i, y_i, t_i)$，其中$t_i = t_{i-1} + \Delta t$，这些点构成了一个拦截点点集P. 注意到为了比较过程中的能量消耗，需要先对每一个点P_i搜索出一条简单的可行路径，得到路径D_i后再针对D_i计算相应的加速度平方和。

路径规划属于自动驾驶中的决策部分，是自动驾驶技术的核心模块之一，一个好的路径规划策略可以决定车辆行驶路线的优劣和流畅程度，如何在短时间内找到一条路程短、效率高的行驶路径是路径规划的目标。路径规划的相关算法有很多，总体上又分为全局路径规划和局部路径规划。前者也称作静态规划，此类规划算法的前提是系统要知道全部的环境信息，包括环境内所有障碍物的分布情况，再从起始点规划出到目标点的路径；后者也被称作动态路径规划，系统并不知道全部环境状况，而是在车辆运动过程中，通过搭载的传感器实时记录观测到的环境信息，从而构建出其所处位置的障碍物情况，再进行路程规划。相较而言，局部路径规划需要较高的技术支撑。本实验中用到的规划算法均为全局路径规划。

而在全局路径规划的方法中，最主要的几类包括传统路径规划算法（Dijkstra、A*等）、智能优化算法（遗传算法、机器学习方法等）还有将传统和智能结合起来的方法。表 3-1 中列举了几种代表性的方法原理和优缺点比较[2]，对比之后选用三维 A*做初步的路径搜索，以辅助拦截点选择。因此，本小节主要介绍 A*搜索及其扩展到三维空间的实现，其他方法将在 3.3 中的路径规划部分详细介绍。

表 3. 1 几种主要路径规划方法对比

方法	主要原理	优点	缺点
Dijkstra	以起点为中心，向外扩展搜索，直至目标点	一定能找到最短路径，且可以一次搜索多个目标点	遍历节点多，不适用于自动驾驶路径规划
A*	启发式搜索，设置了估计函数	与 Dijkstra 相比搜索效率提高	

人工势场法	综合障碍物的排斥力和目标点的引力进行避障和规划	计算量偏小，效率较高	容易陷入局部极小
RTT	基于快速扩展随机树	快速有效，能处理高维复杂环境下的搜索	搜索结果可能不是最优的
遗传算法	主要流程包括种群初始化，适应度函数计算，选择、交叉和变异	不会陷入局部最小，多用于局部路径规划	计算量大，高维情况难以处理
神经网络	用神经网络描述所处环境约束	端到端，可解决任意复杂非线性问题	需要前期进行训练，且可能不收敛

A*是一类启发式的搜索算法，其核心在于启发函数的设置，这里简要介绍算法流程。

首先，需要对待搜索的区块进行间隔采样，将整个区域划分为相同大小的正方形栅格，用二维数组表示，数组每一项的值代表这一块区域“可行”或“有障碍”，令方格的中心为搜索“节点”，则算法目标是从起始节点 A 出发，依次检查当前节点周围的可行节点，不断向外围扩张，直至找到一条可行路径，到达目标节点 B。完成了上述对搜索区域的采样和量化后，正式开始查找最短路径，建立一个存储节点的开放列表 open list，作为待检查的节点列表，首先将起始节点 A 放入，其次，查看与起始节点相邻的方格节点，若这些节点是“可行”的，则将其存入 open list 中，并将起始节点 A 设为这些可行节点的“父亲”（parent node）。检查完节点 A 的全部相邻节点后，需要将 A 点放入一个同样用于存储节点的闭合列表 close list 中，close list 中都是已经不再需要关注的节点。

接着需要从 open list 中选取估计函数值 f 最小的节点，这里有等式$f = g + h$，其中 g 代表了从起始节点到当前节点的移动代价（距离），而 h 是启发函数，代表了该节点到目标节点的估算距离，这个值不是到目标的真实距离，而仅是估计值，通常使用曼哈顿距离来计算 h 的大小。计算出 open list 中每个节点的 f 值后，选取最小的那个点，重复上述对起始节点 A 的操作，即检查相邻的节点，“可行”的放入 open list 中，若是“障碍”或已经在 close list 中的忽略即可，最后将当前节点转入 close list，再次遍历 open list，不断进行上述操作，直到目标节点被加入到 open list 时停止，路径查找成功；或者没有找到目标节点，并且 open list 为空，则查找失败。若成功找到路径，只需要从目标开始沿着指向“父亲节点”的指针

一路找回到起点，就可以得到搜索出的最短路径了。A*算法的整体流程图如图 3.1 所示。

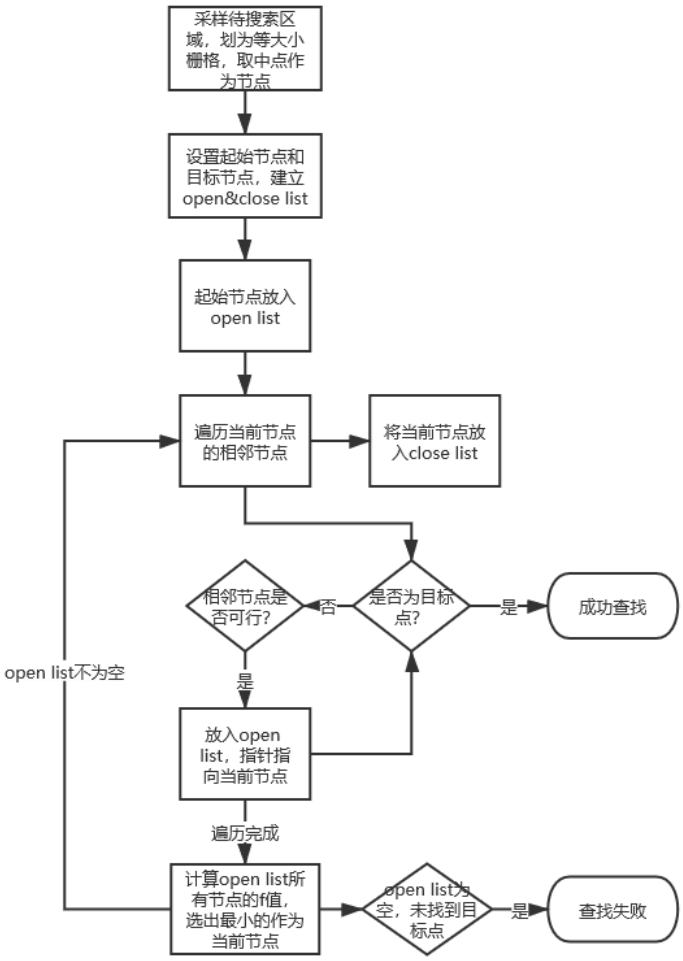


图 3. 1 A*算法流程

在本实验中，考虑到可能存在动态障碍物的情况，结合目标点位是三维数组(x_i, y_i, t_i)，将 A*算法扩展到三维情况下，总体思路不变，每次待检查的相邻点位从 8 个增加到 17 个（这里时间维度上无法倒退，需要除去一些相邻点）。同时，为了满足小车的运动学约束，应检查采样间隔是否符合要求，即节点间的距离除以间隔时间要小于小车行驶的最大速度，距离除以时间的平方要小于最大加速度等，根据需要及时调整或重新采样。

针对每一个拦截点P_i搜索出路径D_i后，将路径上一系列节点(x_{ik}, y_{ik}, t_{ik})转换成每个点距离起点的里程数，得到一组里程数向量$(s_{i0}, s_{i1}, s_{i2}, \dots, s_{in})$，其中$s_{i0} = 0$，再用差分法求每一段路径上的速度和加速度，公式如下：

$$v_{ik} \approx \frac{s_{i(k+1)} - s_{ik}}{\delta t}$$
$$a_{ik} \approx \frac{s_{i(k+2)} - 2s_{i(k+1)} + s_{ik}}{\delta t^2} \tag{3-2}$$

得到路径D_i上的加速度$\left(a_{i0}, a_{i1}, \dots, a_{i(n-2)}\right)$后，结合式 3-1，小车在拦截过程中的能量消耗可以表示为$J = \sum_{k=0}^{n-2} a_{ik}^2$，又有$T = t_i$，可以求出每一个拦截点的目标函数值，比较后取函数最小值的点作为最优拦截点，从而完成该部分决策。

3.3 轨迹规划

经过 3.2 节介绍的最有拦截点选取，已确定了拦截点(x^*, y^*, t^*)，轨迹规划的目标即从小车当前位置规划出一条路径到(x^*, y^*)，总耗时为t^*，要求避开环境中的障碍物，同时路径要尽量光滑，满足车辆运动学和动力学特性。

在自由空间中做车辆的运动规划需要用更有效的方法来解决“开放式”环境带来的诸多问题。如车辆存在非完整的动态约束，因为在自由空间中，小车可能会以接近其物理极限的方式运动；又如车辆需要精准地避开障碍物，当遇到间隙很窄的障碍物时，能成功找到狭窄但可通过的路径；还有计算实时性以及驾驶过程中舒适性的问题。这些问题使开放空间的路径规划成为自动驾驶领域一个颇具挑战性的话题。

以是否将路径和速度解耦为判断可以将规划方法分为两大类。一类方法把路径和速度耦合处理，同时解决路径和速度的规划和优化，其本质是一个存在等式和不等式约束的优化问题，优化目标是路径中能量消耗少、舒适度高、距离障碍物远等，理论上可以用传统的最优控制方法求解。这里简单介绍最优控制中的极小值原理。设受控系统的状态方程为$\dot{x}(t) = f[x(t), u(t), t]$，其中$x(t)$是系统状态量，在小车中即二维平面上的位置和两个方向的速度，$u(t)$是控制量，自动驾驶中即油门量和方向盘转角。系统初始状态$x(t_0) = x_0$，终端状态$\mathrm{x(t_f)}$满足约束$g\left(x\left(t_f\right), t_f\right) = 0$，寻求最优控制$u(t) \in U$，使设定的性能指标$J = \Phi\left(x\left(t_f\right), t_f\right) + \int_{t_0}^{t_f} L[x(t), u(t), t]dt$取得最小值，性能指标中可以通过惩罚加加速度（jerk）的累

积项来确保驾驶过程中的舒适性，通过惩罚加速度和速度项来实现能量消耗最小化的目标。

将路径规划和速度规划一并解决是很理想的方式，但也存在诸多明显的缺陷。首先，该类方法要在程序中求解包含多个等式、不等式的优化问题，这导致了庞大的计算量，且运算量会随着环境障碍物变复杂而越来越大，最终可能无法满足车辆驾驶的实时性要求；其次，该类方法鲁棒性不足，若目标点出现微小遍动，控制策略即宣告失败，而且当环境中存在障碍物时，整体优化问题可能非凸，需要考虑凸近似等解决方法，使算法进一步变得复杂。

因此，本实验选取将路径规划和速度规划解耦的方法，保证了运算的高效性，并尽量满足路径的平滑，以及速度规划的可行性和舒适性。下面将分别介绍路径规划和速度规划所采用的方法。

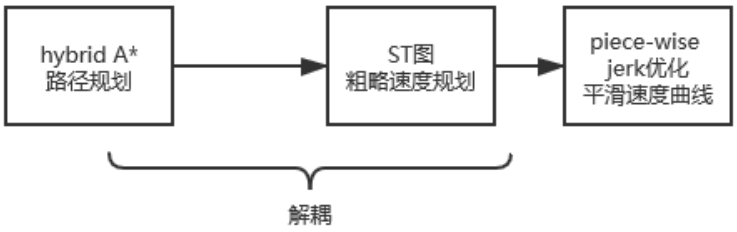


图 3. 2 轨迹规划流程

3.3.1 路径规划

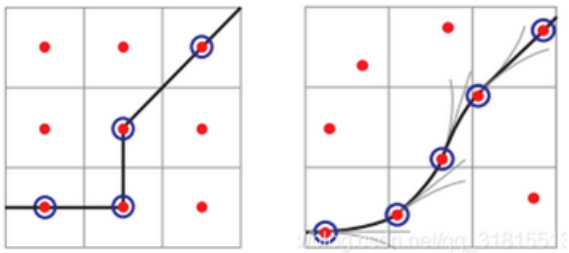
前文在 3.2.2 小节中大致介绍过路径规划及其相关算法，各种算法的原理和优缺点比较见表 3-1，这里详细介绍几种常见的路径规划算法。首先广度优先搜索（BFS）是一种能够找最短路径的方法，但 BFS 的算法效率不高，占用空间较大，不适用于本项目。Dijkstra 算法可以一次获得从起始点到多个目标点的最短路径，针对每个点记录从起点到它的最短路径长度，之后再遍历到这个点时，由于该点已经记录了以前的最短路径，只需比较当前路径和记录的路径哪个花费更少，并更新即可。A*算法在前面已有详细的讲述，它利用了启发式的特点，有方向地进行搜索，从而相较 Dijkstra 提高了效率，与 A*比较相似的还有 D*搜索算法，D*也是一种启发式算法，但与 A*不同的是，它是以目标点为起始开始搜索的，直到机器人当前所在节点从 open list 中出队为止，如果过程中某节点有所改变，算法会重新寻路，因此是一个动态寻路算法，适合于应对周围环境未知或动态变

化的场景，效率与 A*相差不大。RRT 算法[30]全称是快速探索随机树，它抛掉了对空间环境的建模过程，直接对采样到的点做碰撞检测，将搜索引导至没有障碍物的空白环境里，从而快速、高效地处理在高维空间和具有复杂约束场景下的路径规划，但该算法概率上完备且不最优。

此外，上述传统的路径规划算法均存在一个问题，即规划出的路径不够平滑，可能不满足车辆行驶的运动学和动力学特性，因此这里选用混合 A*算法（hybrid A*）[29]，该算法总体流程与 A*相似，不同之处对比见表 3-2。可以看到混合 A*在 A*节点的基础上加了一个θ维度，即将车辆的姿态角度也考虑到算法当中，还可以设置参数 R，表示车辆是向前行驶还是向后倒车，由于加入了θ角，混合 A*中节点不一定处在栅格的正中央，且节点间的连接相对平滑，如图 3.3(b)所示；而 A*的节点单纯是二维平面坐标点，因此节点间的连接僵硬不平滑，如图 3.3(a)。此外，在提出混合 A*的论文中，启发函数用的是当前点到目标点的 Reeds-Shepp 距离，实际情况中，在线计算这样的启发函数会消耗大量的计算时间，论文中为了提高效率提前储存了地图中每个点到目标的 Reeds-Shepp 距离，因此本实验中编写的混合 A*算法还是沿用了 A*的曼哈顿距离。

表 3. 2 混合 A*与 A*的对比

算法	Hybrid A*	A*
搜索维数	(x, y, θ)或(x, y, θ, R)	(x, y)
启发函数	Max(Reeds-Shepp, A*)	曼哈顿距离
节点	车辆运动学模型	二维平面坐标
节点连接	平滑，节点可以不在栅格正中间	不光滑，节点是栅格中点
缺点	不具完备性	不满足车辆运动学特征



(a) (b)

图 3. 3 混合 A*与 A*节点图示对比

混合 A*的流程图如图 3.4 所示，基本步骤和 A*相似。首先初始化起点、终点，确定车辆模型，建立 open list 和 close list。接着将起点放入 open list，扩展其相邻节点并设置指针，计算 open list 中每个节点的 f 值，再选出 f 值最小的一个作为当前节点，重复上述过程，直到扩展至目标节点。为了加速算法效率，在接近目标点时停止搜索，选择用 Reeds-Shepp 曲线连接当前点和目标，此处也可以用 dubins 曲线连接，减少搜索的次数。

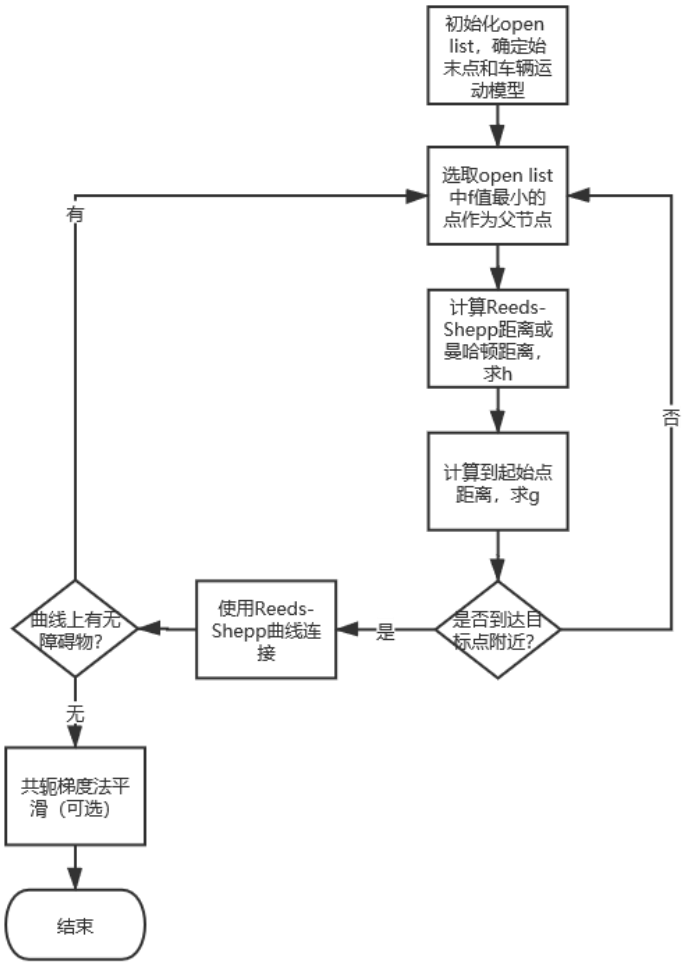


图 3. 4 混合 A*流程图

3.3.2 曲线平滑

单纯使用混合 A*搜索仍可能存在路径不够光滑的问题，图 3.4 的流程中提到，在算法的最后一步可选择用共轭梯度法对路径做进一步平滑。共轭梯度法是一种迭代式的优化算法，仅需利用函数的一阶导数信息，所需计算空间小，且具有易收敛、稳定性高的优点。假设当前已有一系列待平滑的离散路径点x_i，令所求路径为y_i，则需要优化的目标为：

$$\min(x_i - y_i)^2 + \alpha(y_i - y_{i+1})^2 \tag{3-3}$$

运用梯度下降的思想优化目标，迭代求取最优解，具体步骤如下：

$$\begin{aligned} y_i &= y_i - \alpha(x_i - y_i) \\ y_i &= y_i - \beta(2y_i - y_{i-1} - y_{i+1}) \end{aligned} \tag{3-4}$$

在 MATLAB 中实现上述算法，用一段简单的不平滑路径进行测试，其中$\alpha = 0.5, \beta = 0.2$，当迭代到曲线基本不发生变化即停止。结果如图 3.5 所示，其中蓝色虚线是初始路径，红色点线是平滑后得到的结果，可以看到经过算法处理后，曲线平滑程度明显提高，有利于提升车辆行驶的可行性和舒适度。

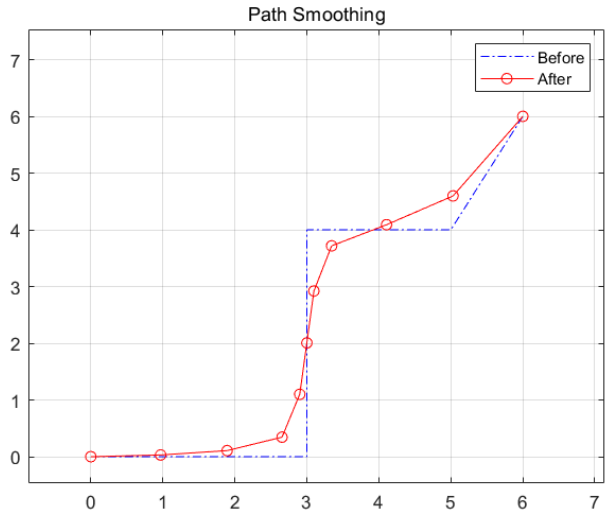


图 3. 5 共轭梯度平滑

3.3.3 速度规划

除了路径规划，速度规划也是非常重要的一个部分。在当今自动驾驶任务中，系统对车辆的速度规划有以下几点要求。一是要具备灵活性，即车辆能够在较为复杂、拥挤的城市环境中行驶，灵活避开多个动态障碍物；二是速度的分配要尽可能平滑，平稳变化的速度、加速度是驾驶舒适性的前提，并且速度规划要结合路径的几何特点，在路径曲率不同的地方分配合适的速度，如曲率较小的路段速度应尽量小，而曲率较大的地方速度也可以变快，提高整体舒适感；三是要遵循车辆运动学特性，满足车辆最大速度、加速度、加加速度的约束条件，包括特定环境中对速度上下限的要求，以使车辆在现实中可以执行规划出来的速度。

本实验借鉴百度 Apollo 平台的速度规划方法[25]，采用离散启发式速度规划和加加速度分段速度平滑结合的方法来处理速度规划的问题。首先用离散的启发式搜索粗略规划出一个速度曲线，这里需要用到路径-时间障碍图（即 ST 图）作为分析工具，这类图横轴坐标为时间，纵轴坐标是路程，选定一条路径后，将各种静态和动态障碍物映射到 ST 图上，ST 图上障碍物的投影表示车辆会在 t 时刻 s 处与障碍物发生碰撞。

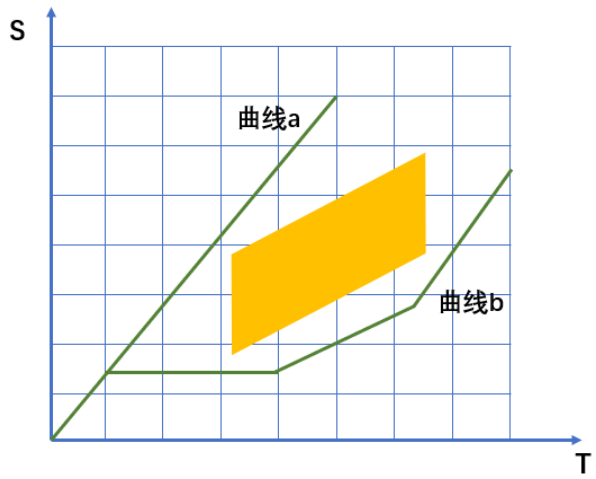


图 3. 6 ST 图规划

如图 3.6 所示，ST 图将时间轴和沿着路径计算的累计距离离散化，生成等时间间隔和等距离的网格坐标，坐标系中黄色平行四边形部分是障碍在 ST 图上的投影。根据一系列车辆行驶中需要的考虑因素，设置代价函数，赋予每个网格一个代价值，同时两个网格之间的边界也建立代价，用来做加速度相关的约束，之

后利用图搜索的策略，找到代价最小的一条路径。图中标出了两条用启发式搜索可能得到的规划路径，其中速度曲线 a 表示保持高速度超越障碍物，而速度曲线 b 表示车辆在中途减速停车，避让了障碍物。

用启发式的方法得到一条粗略的速度规划曲线后，需要用分段加加速度平滑算法（piece-wise jerk）对速度曲线做进一步的处理，得到可以被控制器直接使用的光滑$s(t)$曲线，提高速度分配的平滑性。该模块的输入是搜索得到的一系列(t_i, s_i^h)点，输出是在加速度级别连续、平滑的函数s(t)，设置时间间隔Δt，将$s(t)$离散化，则该算法中需要求解的变量为

$$\left\{s_i,\ \dot{s}_i\left(\frac{ds}{dt}\right),\ \ddot{s}_i\left(\frac{d^2s}{dt^2}\right)\right\}$$

分别对应每个点的路程、速度和加速度，并且假设在相邻的两个点之间，加加速度是常数，即$\dddot{s}_{i\to i+1} = (\ddot{s}_{i+1} - \ddot{s}_i)/\Delta t$，这也是算法名称 piece-wise jerk 的由来。

平滑的目标是尽量使位置变量s_i贴近先前粗略规划出的s_i^h，并且保证车辆在行驶过程中的舒适性，因此需要在目标函数中对加速度和加加速度进行惩罚，优化目标函数设计如下：

$$f = w_h * \sum_{i=0}^{n-1}\left(s_i - s_i^h\right)^2 + w_{\ddot{s}} * \sum_{i=0}^{n-1}\ddot{s}_i^2 + w_{\dddot{s}} * \sum_{i=0}^{n-2}\dddot{s}_{i\to i+1}^2 \tag{3-5}$$

约束条件包括速度分配要连续，并且规划的加速度和加加速度要在车辆可行的范围内：

$$\begin{cases} \dot{s}_{i+1} = \dot{s}_i + \ddot{s}_i * \Delta t + \frac{1}{2}\dddot{s}_{i\to i+1} * \Delta t^2 \\ s_{i+1} = s_i + \dot{s}_i * \Delta t + \frac{1}{2}\ddot{s}_i * \Delta t^2 + \frac{1}{6}\dddot{s}_{i\to i+1} * \Delta t^3 \end{cases}$$

$$\begin{cases} \ddot{s}_i \in [\ddot{s}_{i_{min}}, \ddot{s}_{i_{max}}] \\ \dddot{s}_{i\to i+1} \in [\dddot{s}_{min}, \dddot{s}_{max}] \end{cases} \tag{3-6}$$

由式 3-5 和 3-6 可以看出，这是一个有等式和不等式约束的二次规划（QP）问题，二次规划问题有多种求解方法，目前工程上常用的是 OSQP，一种基于

ADMM 算法的 QP 问题求解器[26]，本实验中也使用了 OSQP，求解用时在毫秒数量级。

上述先用启发式速度规划，再进行速度曲线平滑的方法有优点也存在一定待优化的问题，优点在于该算法较好地解决了路径位置 s 与时间 t 之间相互耦合的问题。而若直接使用分段加加速度的方法求解速度控制，跳过 ST 图搜索的步骤，则会出现车辆位置和时间相互影响的情况，比如车辆位置发生变化时，与位置相关的约束条件，包括限速情况、路径的曲率大小等都会随之变化，而到达目标点的时间也会变化；又如要求的到达时间发生变化时，会影响速度的分配，从而车辆在某时刻的位置也会改变。初始在 ST 图的粗略规划可以将位置和时间解耦，将时间 t 视为常量。但 ST 图搜索中离散化操作影响了得到路径的质量，不能准确反映车辆的运动，并非最优的速度规划。

综上所述，本章节主要介绍了决策规划模块的整体流程，包括最优拦截点选择和从起始点到拦截点的轨迹规划，轨迹规划又解耦为路径规划和速度规划。文中详细阐述了每一步用到的算法思路和理由，同时对比了其他相似算法，证明了本文所选算法更高效、更鲁棒。

第4章 实验平台与仿真环境

4.1 实验平台介绍

本实验主要通过仿真的方式开展，因此本章将介绍所使用的实验平台和自主搭建的仿真环境。实验选用了开源的机器人操作系统 ROS（robot operating system），搭配仿真软件 Gazebo，下面分别对两者进行介绍。

ROS 发布于 2010 年，是一个免费、开源的元操作系统，专门用于机器人系统的开发和设计，其实际是一个框架平台，需要依赖其他操作系统，如 Linux 才能运行。ROS 内部支持 c/c++、python 等多种编程语言，并且其系统本身具有轻量级的特点，有大量独立的库做支持。通常来讲，一个机器人系统会由多个模块和传感器共同构成，这些模块或传感器之间需要互相通信才能协作完成一些任务，传统的操作系统在实现进程间通信的过程中会调用、消耗大量的系统资源，而 ROS 为此提供了快速且高效的进程间通信（分布式计算），满足了机器人系统的要求。catkin 是 ROS 用来生成可执行文件、库、接口的构建工具，ROS 中编写的各级文件都存放在用户创建的一个工作空间中，工作空间内有各种功能包，每个功能包中又包括了功能包清单、消息类型、代码、库文件等。

ROS 中最核心的概念是节点间的两种通信机制——话题和服务，这里的节点就是上文提到的进程，即构成机器人的模块或传感器，节点间常用话题或服务机制进行通信。这两种通信机制的比较见表 4-1 和图 4.1，可以看到话题机制通过节点间的发布和订阅完成，这种通信模式是异步、无反馈的，可以在多节点与多节点间建立，有缓冲区可以存放到达的消息队列；而服务通过一个服务器发送消息、多个客户端接收消息实现，是同步、有反馈的，实时性高，但没有缓冲区。在机器人系统的搭建中，应结合不同任务的需要，选择不同的通信机制。

表 4. 1 话题和服务机制比较

	话题（topic）	服务（service）
节点模型	发布者/订阅者	服务器/客户端
同步/异步	异步	同步
节点对应	多对多	一对多

实时性	较弱	较强
接收消息反馈	无	有
队列缓冲区	有	无

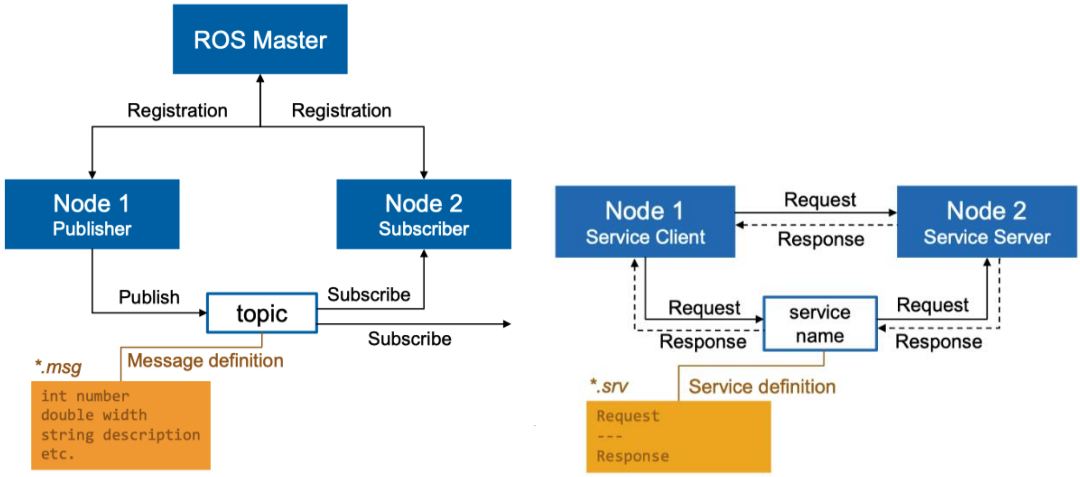


图 4. 1 话题通信（左）和服务通信（右）

Gazebo 是一个常与 ROS 搭配使用的仿真软件，提供专门的 ROS 接口，并内置了接近真实世界的物理引擎，可以在其中构建自定义的机器人运动仿真模型和传感器模型，进行各式各样的运动任务仿真，搭建仿真场景模拟现实中机器人的测试环境，还可以为机器人和仿真环境设置添加各种现实世界的物理特性，如重力、摩擦力等。Gazebo 提供了多种 3D 的刚体动力学仿真工具，可以进行 3D 可视化和人机交互的实验，内涵多种机器人类型和环境模型，同时支持自定义的插件。是一款强大的机器人仿真软件。

4.2 仿真环境搭建

本实验使用 Ubuntu 16.04 版本，安装 ROS-kinetic，用 Gazebo 自主搭建了仿真环境，下面详细介绍仿真环境的设计和搭建。要仿真拦截任务，首先要创建两个智能小车，分别作为移动目标和拦截小车，这里为了简便使用两个相同的模型定义；其次需要创建一个.world 文件生成仿真环境；最后编写 launch 文件用于启动 Gazebo 并导入定义的机器人和环境。

在 Gazebo 中创建自定义机器人，需要自行编写后缀为.xacro 的 XML 文件，并在文件中用 urdf 格式描述机器人组件。首先要定义机器人的“base_footprint”和“base_link”，确定小车主体的坐标系和在世界坐标中的初始位置。其次定义智能小车的四个轮子，包括轮子的宽度、半径、位置和碰撞体积等，再创建关节，将四个轮子连接到车辆主体，之后定义车辆上搭载的激光测距仪和双目摄像头，同样需要确定位置、大小和碰撞体积，并与车辆主体建立关节连接，这些模块后期可以与真实环境中搭载的激光仪和摄像头通信。上述结构均可以使用<material>标签设置颜色和其他材料特性。自定义小车的整体结构如图 4.2 所示。

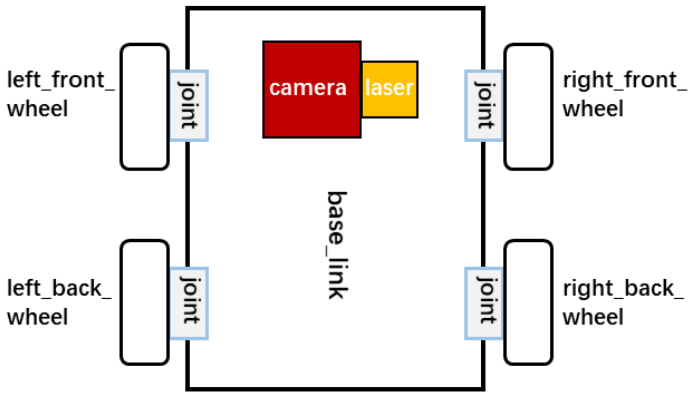
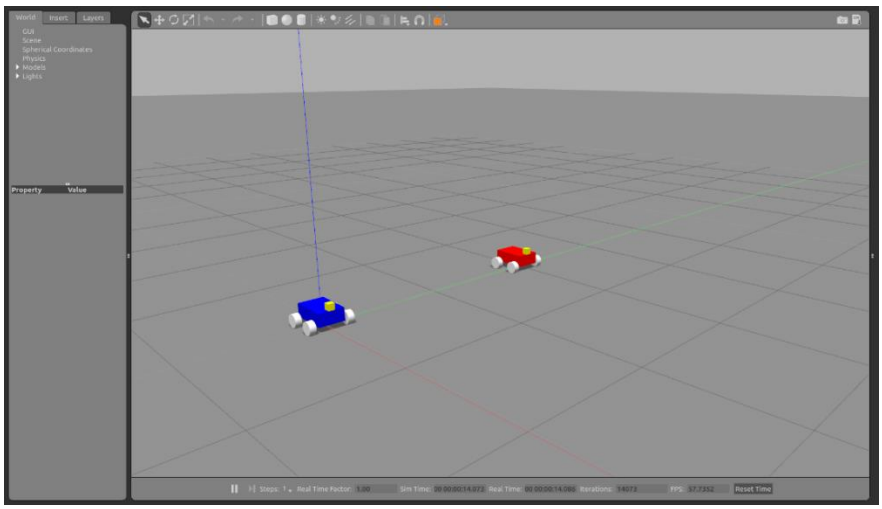


图 4. 2 Gazebo 中小车模型搭建

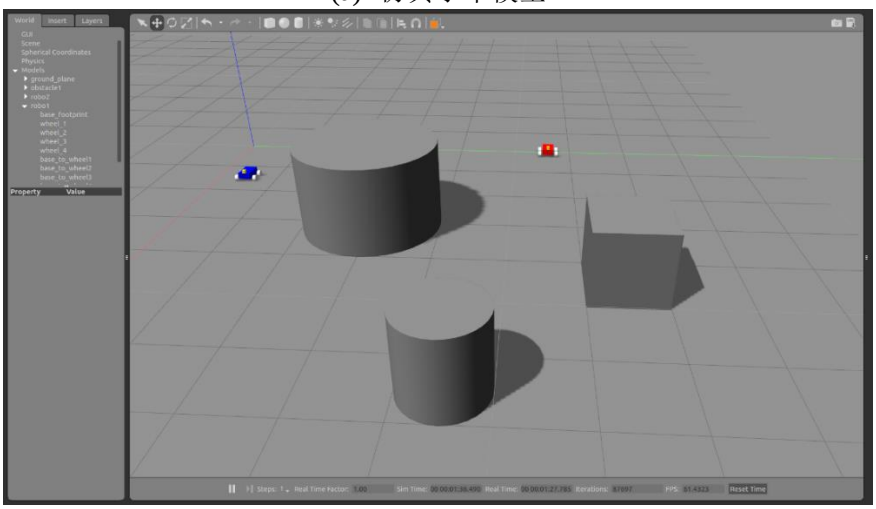
此外，还可以在<Drive controller>标签下设置车辆运动方式，是差速驱动还是滑动转向驱动，本实验中采用滑动转向驱动，这种驱动方式相较差速驱动较简单、易实现，左右两侧的轮子可以按照不同速度、不同方向转动，从而能使机器人获得强大的牵引力进行转向，但该方式下小车里程计会受到滑动影响而出现误差，导致记录不准。

接下来创建 launch 文件夹，编写 gazebo.launch 文件用于启动 Gazebo 仿真环境，加载定义好的机器人模型。launch 文件同样用 XML 语言编写，首先设置一些基本参数，其次加载环境文件，这里可以是 Gazebo 自带的初始仿真环境如 empty_world.world，也可以是用户自己创建的环境，本实验中搭建了带多种形状障碍物的测试场景 obstacle.world。接着加载先前定义的智能小车模型，因为拦截任务中涉及两个主体，因此采用设置类别命名空间（group namespace）的方法区分目标小车和拦截小车，目标小车命名空间是“robo1”，则其所有的参数、模块

名称前都会添加“robo1”的前缀，拦截小车设为“robo2”同理。加载模型的时候还可以设置两辆小车的初始位置和姿态角。用 roslaunch 语句启动 Gazebo 环境，如图 4.3 分别是没有障碍物和存在障碍物时的仿真场景，其中蓝色小车是移动目标，红色小车是拦截小车。至此，仿真环境搭建完毕，可以在 ROS 中编写程序节点，通过订阅 Gazebo 提供的话题/服务进行消息通信和小车控制。



(a) 仿真小车模型



(b) 存在障碍物的仿真环境

图 4. 3 自主搭建的 Gazebo 环境

第5章 实验设计与结果分析

5.1 节点架构

第 4 章中提到了 ROS 中最重要的机制是节点通信，本小节将介绍整个拦截系统的节点架构设计，说明节点间的通信关系。

如图 5.1 所示是用命令行 rqt_graph 绘制的节点通信图，其中椭圆形表示创建的节点，方框表示话题或服务。可以看到用红色方框框出的两个命名空间 robo1 和 robo2，分别代表拦截任务中的移动目标和拦截小车，命名空间下囊括了几个创建机器人自带的节点，包括 joint_state_publisher、robot_state_publisher 等，用户可以通过查看这些节点发布的消息获取小车的实时状态；此外还有常用的 cmd_vel 话题，cmd_vel 话题的话题类型包括三维(x, y, z)的线速度和三维的角速度，创建速度发布节点 velocity_pub 向 robo1 的 cmd_vel 话题循环发送速度指令，控制移动目标按照设定的规律做匀速直线或匀速曲线等运动。

创建轨迹预测节点 traj_estimator 用于预测移动目标的未来轨迹，并运行最优拦截点选取算法，该节点从 robo1 的服务 get_model_state 处获得移动目标在一段时间的位置坐标，并将算法得到的拦截点信息通过服务 chase_ctrl 发送给拦截控

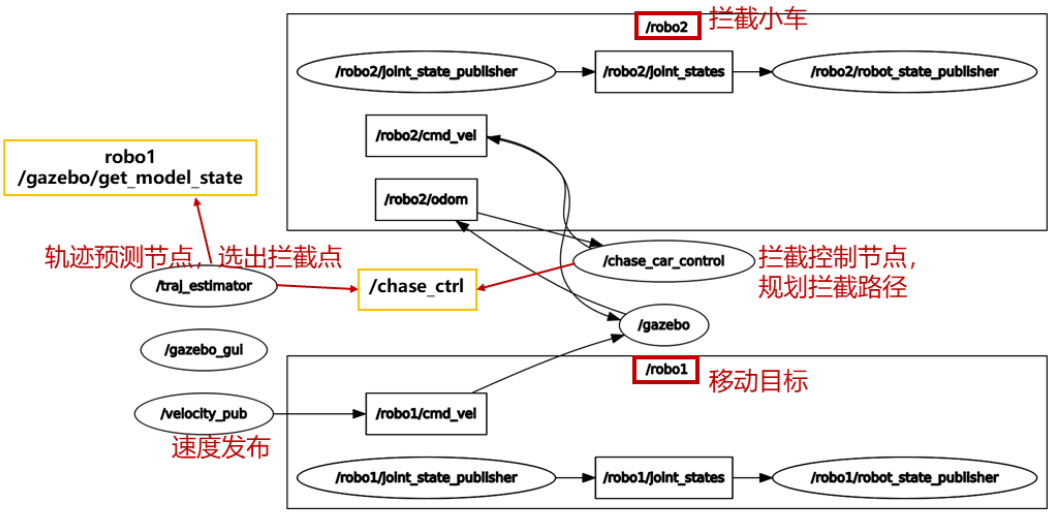


图 5. 1 ROS 节点架构

制节点 chase_car_control。该节点收到拦截点信息后，将拦截点作为目标点，运行轨迹规划的一系列算法，规划出拦截路径和速度，将计算好的速度信息发布给拦截小车 robo2 的 cmd_vel 话题，控制拦截小车避障行驶到拦截点，完成拦截任务这期间节点还会订阅 robo2 的 odom 话题发布的消息，获取 robo2 的里程数据和位置。

以上是本系统的节点设计和通信关系。

5.2 实验结果分析

前文详细介绍了本实验总体的思路流程和选用的方法，下面将对测试得到的实验结果加以分析，考虑到实验的随机性，重复进行实验，每一次所得结果会有一定偏差，均在可接受的范围内。本实验全部代码使用 c++编写，部分模块还改写了 MATLAB 版本，以更好地可视化展示。

5.2.1 预测结果分析

用多项式拟合的方法对移动目标的未来轨迹进行预测，实验中设定移动目标的运动模式为匀速直线运动、匀加速直线运动和匀速曲线运动。对移动目标进行 15 秒的观测，每秒采样 5 次，分别对未来数个时刻的位置坐标进行预测，与未来时刻的真实位置做对比，列表展示如下：

表 5. 1 轨迹预测结果分析

(a) 匀速直线运动（0.1m/s）

时刻t/s	0	15（+0）	20（+5）	25（+10）	30（+15）
预测位置x_p		2.8206	3.3214	3.6827	4.0428
真实位置x_r	1.3042	2.8133	3.3378	3.7972	4.3259
偏差α		0.26%	0.48%	3.09%	6.52%

(b) 匀加速直线运动（0.1m/s, 0.01m/s²）

时刻t/s	0	15（+0）	20（+5）	25（+10）	30（+15）
预测位置x_p		2.6104	3.8647	5.4331	6.8573

真实位置x_r	0.0023	2.6237	3.9789	5.6187	7.3894
偏差α		0.51%	2.87%	3.31%	7.20%

(c) 匀速圆周运动（0.2m/s, 0.02rad/s）

时刻t/s	0	15（+0）	17.5（+2.5）	20（+5）
预测位置(x_p, y_p)		1.9465, 0.8872	2.2252, 1.3544	2.2927, 1.6342
真实位置(x_r, y_r)	-0.5822, 0.1116	1.9507, 0.8912	2.1894, 1.2819	2.3091, 1.7495
平均偏差$\frac{\alpha_x+a_y}{2}$		0.33%	3.64%	3.65%

	22.5（+7.5）	25（+10）	27.5（+12.5）
预测位置(x_p, y_p)	2.2500, 1.9581	2.0451, 2.3311	1.6556, 2.5293
真实位置(x_r, y_r)	2.2837, 2.1945	2.1250, 2.7055	1.9234, 3.0766
平均偏差$\frac{\alpha_x+a_y}{2}$	6.12%	8.79%	15.85%

表中时刻一栏表示小车处在运动中的时间，0 时刻小车在初始位置，运动 15 秒用于记录观测数据，从 15 秒起预测未来位置坐标，直线运动预测到 30 秒，曲线运动预测到 27.5 秒。可以看到，无论移动目标处于匀速直线、匀加速直线或匀速曲线运动时，多项式拟合的方法均能对未来轨迹有大致准确的预测。预测时间在 10 秒以内，该方法可以把位置偏差控制在 10%以下，随着预测时间的增长，误差越来越大，且取相同的预测时间，曲线运动、匀加速直线、匀速直线的误差逐渐减小。同时实验也发现在匀加速直线运动模式下，加速度越大预测误差越大，这符合经验判断。

总体来说，匀速直线运动下预测准确率最高，本实验中设定的预测时间上限为 15 秒，在这段时间内产生的误差都是可以接受的，也可以考虑通过增长观测目标运动时间进一步提高准确率。后续选取最优拦截点的步骤中，有$t \in [0,15]$，在这条预测轨迹上等间隔选取 N 个点作为候选拦截点。

5.2.2 拦截点选取结果分析

实验中取 N=5 个等时间间隔的拦截点，假设移动目标做曲线运动的情况，得到 5 个候选拦截点后，将地图按合适的横纵距离采样离散化，保证每个候选点都单独在一个栅格内，检查障碍物和边界区域，并确定拦截小车的起始栅格，完成 A*算法的初始化。本实验中采用三维 A*搜索以应对存在动态障碍物的情况，但

三维 A*的搜索结果不便可视化，这里展示在二维坐标平面上的投影。如图 5.2 所示，红色五角星表示拦截小车的起点，黄色圆点表示五个候选拦截点，从点 1 到点 5 分别对应$t = [3{:}\,3{:}\,5]$，对每个点做三维 A*搜索，在保证满足车辆最大速度、最大加速度约束的情况下，得到每个点对应的路径，并用差分法计算出每条路径上的能量损耗，结果如表 5-2 所示，表格中能量消耗一栏列出了每条路径上的加速度平方和[1]，对比可知从起点到拦截点 3 的搜索路径能量消耗最少，也即最平滑，车辆驾驶舒适度最高。按一定比例加上拦截时间后，拦截点 3 的目标函数值最小，即点 3 为最优拦截点。图 5.2 绘制出了从起始点到点 3 的搜索路径，可以看到该路径较为平滑、拐点少，基本符合能量消耗最少的预期。

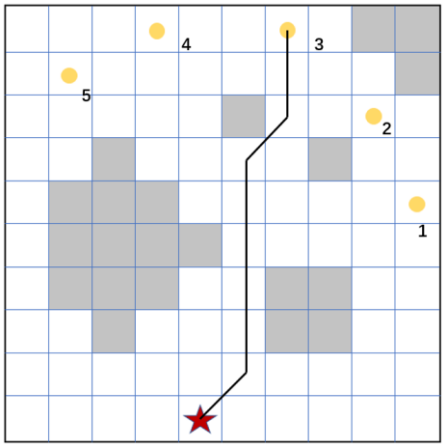


图 5. 2 A*算法搜索结果

表 5. 2 最优拦截点选取

拦截点	1	2	3	4	5
拦截时间(s)	3	6	9	12	15
能量消耗	35.14	36.82	6.863	8.578	41.72
总代价	38.14	42.82	15.863	20.578	56.72
搜索时间(s)	0.3292	0.2351	0.0677	0.1352	0.2465

[1] 此处能量消耗不是准确值，仅反应相对大小，可以进行等比例扩大或减小

此外，表 5-2 中还列出了每个点的三维 A*搜索时间，测试中将时间维度也等分为了 10 份，可以看到对于$10 \times 10 \times 10$的地图，搜索时间基本在几百毫秒的数量级，而点 3 的搜索时间更是仅有几十毫秒的数量级。试想如果不使用前文提到的分层优化的思想，省略利用 A*搜索的粗略结果选取拦截点步骤，而直接对每个候选点做后期的轨迹规划，再计算能量消耗，会花费大量的计算资源和数秒级别的运算时间，这对有实时性要求的系统是不可接受的。但需要注意的是，三维 A*的搜索用时可能随着采样空间的扩大几何倍数增长，因此在粗略搜索的这一步应尽量避免太密集的采样，$10 \times 10 \times 10$空间是比较合适的选择。

5.2.3 轨迹规划结果分析

确定了拦截点后，使用混合 A*算法规划出一条从起点到拦截点的路径，该路径较为平滑，且满足车辆的运动学特性，下面通过一系列测试结果展示混合 A*的性能。

如图 5.3 是无障碍物时的混合 A*规划结果，算法输入的地图是一张 400*400 像素大小的图片，该场景下起点是$(150, 40, 1.57)$，目标点是$(300, 350, 0)$，图片左下角为坐标原点$x = 0, y = 0$，其中$\theta = 0$为车头朝右，$\theta = 1.57$车头朝上，朝向角以此类推。算法还需输入车辆的性能参数，本实验中设定的小车轴距 wheel_base=0.787，最大转向角 max_steer=1.04，由图 2.8 所示的小车模型，可以通过轴距和最大转角求得最小转弯半径，限制车辆的运动路径。当环境中不存在障碍物时，小车调整角度从起点直线行驶到目标点附近，并通过 dubins 曲线[2]连接到目标，生成最后一段路径。

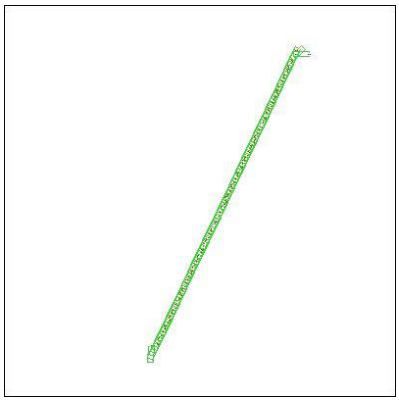


图 5. 3 无障碍物的混合 A*

[2] 也可以采用 Reeds-Shepp 曲线，加入车辆后退的情况

场景中加入障碍物后，规划的路径出现变化，如图 5.4 中灰色部分为障碍物，相比图 5.3 中的路径，5.4 中先保持$\theta = 1.57$的朝向直线行驶一段距离后，再转向朝目标点行驶，从而避开障碍。

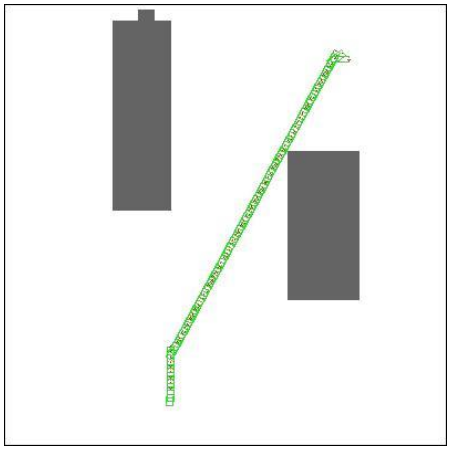
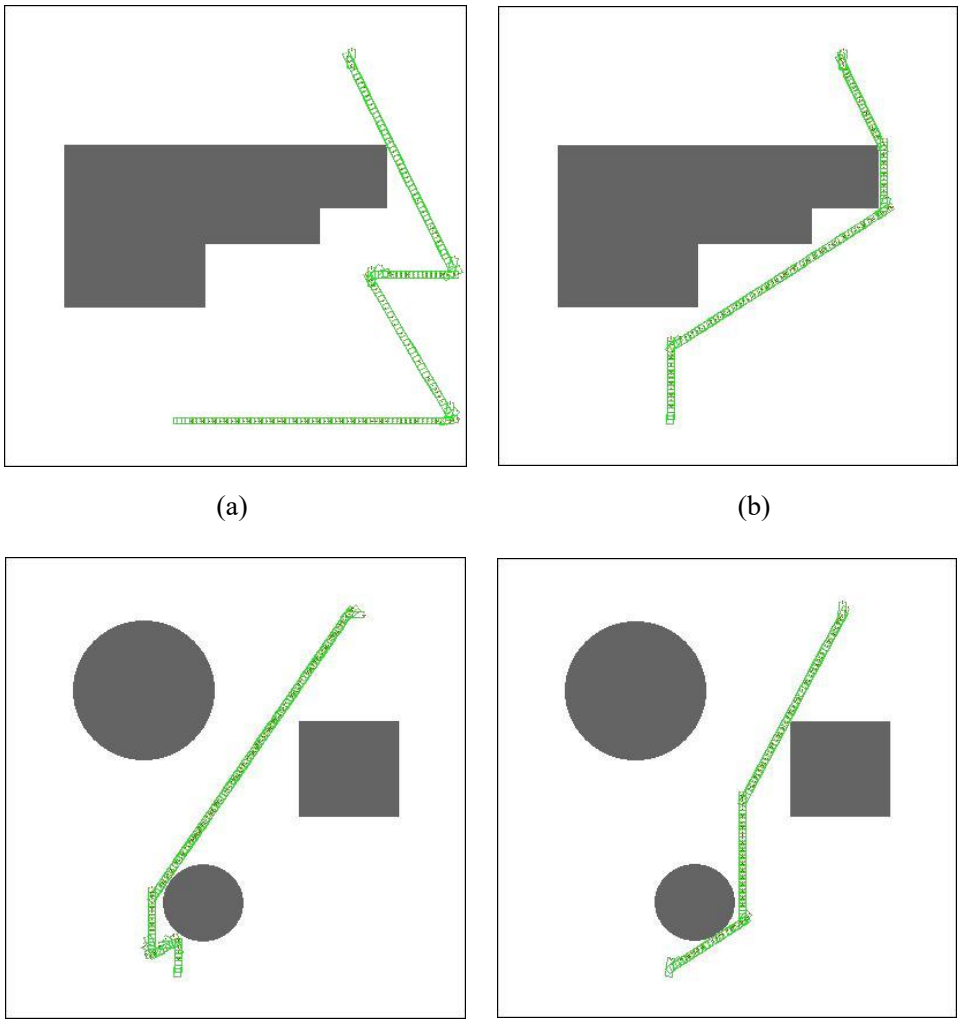


图 5. 4 存在障碍物的混合 A*

实验中还对混合 A*设置了一些参数，包括对转向的惩罚因子，调大该系数后路径规划过程中会尽量减少转向，保证路径的平滑，但也不能太大，否则将导致寻路失败。此外还有参数 step_size，前文对 A*和混合 A*的算法介绍中均有提到，在遍历 open list 表中节点时，要对每一个点进行估值，求得该点的 f 值大小，而 f 由该点到目标的曼哈顿距离 h 加上起始点到该点的真实距离 g 得到，每一个放入 open list 里的点都有其父节点，这里的 g 可以由其父节点的 g 加上一个 step_size 得到，反映到本实验中 step_size 即车辆每一步走的距离。

调节参数 step_size 进行对比实验，如图 5.5 所示是在两组不同的障碍物环境下，step_size 不同得到的路径规划结果，其中图（a）和（c）取 step_size=1，图（b）和（d）取 step_size=0.5，可以看到在步幅设置较大时，在 f 值中从当前点到起始点的真实距离 g 占据较大比例，削弱了启发函数 h，导致算法启发式特征不明显。如（a）和（c）中，规划出的路径基本都是遇到障碍物或边界才转弯，没有向着距离目标点靠近的方向扩展节点，导致最终得到的路径长度较长，且非常不平滑，有明显的大幅度转向；而缩小了步幅参数后，如（b）和（d），启发式的特点就明显展现了出来，搜索出的路径基本为最短路径，且整体看来较为平滑，

没有呈钝角的转弯，车辆行驶过程中的舒适度大幅提高。因此选取合适的 step_size 值是非常重要的。



(a) (b)

(c) (d)

图 5. 5 不同 step_size 下的规划路径

实验同时分析了混合 A*的算法效率，由于加入了车辆转向角θ维度，混合 A*的计算开销较大，算法消耗时间的多少取决于规划过程中共扩展了多少节点，表 5-3 中列出了上述各图片所示结果的搜索时间，结合不同场景下的障碍物情况，可以看到在无障碍和障碍物较稀少时，搜索效率很高，总搜索时间在毫秒级，而当障碍物较多或占据了大片区域时，路径长度增加，扩展的节点数增

多，搜索效率下降的很快，用时可达数秒。但通常情况下，目标距离起始点较劲，算法不会耗时太久，实验中可以加入时间约束参数，搜索时长超过某个时间上限后，即放弃本次路径规划，重新预测移动目标轨迹，开始新一次的拦截过程。

表 5.3 混合 A*搜索效率

	无障碍	图 5.4	图 5.5（b）	图 5.5（d）
搜索时间（s）	0.0173	0.0358	2.7036	0.2847
路径长度（节点数）	101	141	584	213

本章针对实验中用到的各个算法进行了仿真测试，展示、分析了实验结果和计算效率。在自主搭建的仿真环境上部署本文算法，成功实现了对移动目标的拦截任务，结果图如图 5.6 所示。

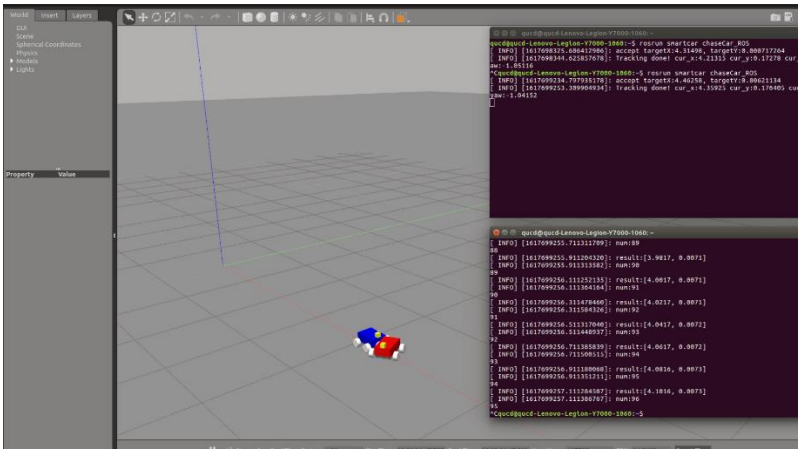


图 5.6 成功实现移动目标的拦截仿真

第6章 总结与讨论

6.1 研究总结

本文主要讨论了移动目标的轨迹预测和拦截问题，设计了一套完整的算法流程，并利用仿真软件自主搭建了仿真环境，对算法进行测试，成功实现了对移动目标的拦截任务。

算法包括感知、预测、决策规划与控制四个部分。首先通过传感器观测移动目标，采样记录目标在一段时间内的位置坐标，存储为历史移动数据，接着对记录的历史数据进行多项式拟合，利用最小二乘法拟合得到多项式的系数，代入未来时刻 t，求出 t 时刻对应的位置坐标。预测出未来轨迹后，在未来轨迹上等时间间隔选取数个坐标点作为候选拦截点，通过选取目标函数值最小的点来确定最优拦截点。目标函数的设置考虑了拦截时间短和拦截过程中能量消耗少两个方面，对每一个候选拦截点做三维 A*搜索，其能量消耗由路径中加速度平方和求得。选定了拦截点后，将小车当前所在位置作为起点，拦截点作为目标点，进行轨迹规划。采用解耦的轨迹规划方法，即分成路径规划和速度规划两个步骤完成。路径规划选用混合 A*搜索算法，可以保证满足车辆的运动学特性，获得可行的、平滑的路径曲线；速度规划中先用 ST 图启发式搜索得到一条粗略的速度曲线，再通过分段加加速度的方法对曲线进行平滑和速度分配，以提高车辆驾驶的舒适性、安全性。

本文设计的拦截算法可以应用于具有不定轨迹的移动目标在包含静态、动态障碍物环境下的拦截问题，相较于已有的诸多算法适用范围更广、更一般化。同时，本文在多个环节利用“分层”的思想提高了算法效率，如在决策规划模块，采用了先粗略搜索路径，选定拦截点后再做细致轨迹规划的策略，若直接对每个候选点进行轨迹规划则会花费大量运算时间，大大降低算法效率；轨迹规划部分对路径和速度进行解耦，提高了算法效率，增强了算法鲁棒性。

6.2 不足与展望

本文的不足主要有以下两个方面：

1. 对移动目标的轨迹预测不够准确，多项式拟合在未来 10-15 秒以内预测的准确程度较高，但对于更远时刻的位置估计其精确度快速下降，易出现较大偏差，限制了单次拦截任务的时长和拦截点的位置选择。若能够在保证准确率的条件下预测较长时间段的未来轨迹，可能会有更优的拦截点供选择。

2. 本文仅在仿真环境下验证了算法可行性和效率，并未在现实场景中进行测试，现实环境中可能存在诸多噪声和干扰，算法的鲁棒性易受到影响，另外现实中车辆的性能和参数相较仿真中简化模型会更加复杂，部分算法细节可能需要进一步的调整。

在本文的基础上，算法的效率和鲁棒性还可以进一步提高，可以想见在较为稳定地完成对移动目标的预测和拦截后，该算法可以扩展到三维空间，适用于无人机等智能体拦截。若移动目标不再是单一智能体，而是由多个相似智能体组成的、有结构和组织的群体时，如机器人编队，如果能够识别出该编队的运动模式，同样可以通过改进本文设计的预测与拦截算法，实现对编队的攻击。此外，若移动目标不再保持简单、封闭的运动模式，而加入了与环境的交互，对于外界的拦截或攻击会做出一定的反馈，则需要观测者探索学习其反馈机制，从而更好地进行拦截，这也是值得深入的研究方向。

插图索引

表格索引

参考文献

[1] AMiner:2018 智能机器人研究[J].自动化博览,2018,35(10):90-93.

[2] Shen S, Mulgaonkar Y, Michael N, et al. Vision-Based State Estimation and Trajectory Control Towards High-Speed Flight with a Quadrotor[C]//Robotics: Science and Systems. 2013, 1: 32.

[3] Lee T, Leok M, McClamroch N H. Geometric tracking control of a quadrotor UAV on SE (3)[C]//49th IEEE conference on decision and control (CDC). IEEE, 2010: 5420-5425.

[4] Mellinger D, Kumar V. Minimum snap trajectory generation and control for quadrotors[C]//2011 IEEE international conference on robotics and automation. IEEE, 2011: 2520-2525.

[5] 宋俊红. 拦截机动目标的有限时间制导律及多弹协同制导律研究[D].哈尔滨工业大学,2018.

[6] 于清华,黄开宏,卢惠民,郭鸿武. 基于双目视觉的足球机器人对三维空间目标的运动估计与拦截[A]. 中国自动化学会控制理论专业委员会、中国系统工程学会.第三十二届中国控制会议论文集（D 卷）[C].中国自动化学会控制理论专业委员会、中国系统工程学会:中国自动化学会控制理论专业委员会,2013:6.

[7] Codevilla F, Müller M, López A, et al. End-to-end driving via conditional imitation learning[C]//2018 IEEE International Conference on Robotics and Automation (ICRA). IEEE, 2018: 4693-4700.

[8] Sallab A E L, Abdou M, Perot E, et al. Deep reinforcement learning framework for autonomous driving[J]. Electronic Imaging, 2017, 2017(19): 70-76.

[9] 田宪科,张科.导弹拦截点计算及其仿真分析[J].飞行力学,2011,29(02):93-97.

[10] 杨子成,鲜勇,李少朋,任乐亮,张大巧.基于学习的中段反导拦截时间和拦截点预测方法[J/OL].北京航空航天大学学报:1-15[2021-05-24].

[11] Belkhouche F, Belkhouche B. On the tracking and interception of a moving object by a wheeled mobile robot[C]//IEEE Conference on Robotics, Automation and Mechatronics, 2004. IEEE, 2004, 1: 130-135.

[12] Meyer Y, Isaiah P, Shima T. On Dubins paths to intercept a moving target[J]. Automatica, 2015, 53: 256-263.

[13] Shkel A M, Lumelsky V. Classification of the Dubins set[J]. Robotics and Autonomous Systems, 2001, 34(4): 179-202.

[14] Reeds J, Shepp L. Optimal paths for a car that goes both forwards and backwards[J]. Pacific journal of mathematics, 1990, 145(2): 367-393.

[15] Manchester I R, Low E M P, Savkin A V. Vision-based interception of a moving target by a mobile robot[C]//2007 IEEE International Conference on Control Applications. IEEE, 2007: 397-402.

[16] Zhu Q, Hu J, Henschen L. A new moving target interception algorithm for mobile robots based on sub-goal forecasting and an improved scout ant algorithm[J]. Applied Soft Computing, 2013, 13(1): 539-549.

[17] Luo R C, Chen T M. Autonomous mobile target tracking system based on grey-fuzzy control algorithm[J]. IEEE Transactions on Industrial Electronics, 2000, 47(4): 920-931.

[18] Luo R C, Chen T M, Su K L. Target tracking using a hierarchical grey-fuzzy motion decision-making method[J]. IEEE Transactions on Systems, Man, and Cybernetics-Part A: Systems and Humans, 2001, 31(3): 179-186.

[19] Chen J, Liu T, Shen S. Tracking a moving target in cluttered environments using a quadrotor[C]//2016 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS). IEEE, 2016: 446-453.

[20] Altché F, de La Fortelle A. An LSTM network for highway trajectory prediction[C]//2017 IEEE 20th International Conference on Intelligent Transportation Systems (ITSC). IEEE, 2017: 353-359.

[21] Mueller M W, Hehn M, D'Andrea R. A computationally efficient algorithm for state-to-state quadrocopter trajectory generation and feasibility verification[C]//2013 IEEE/RSJ International Conference on Intelligent Robots and Systems. IEEE, 2013: 3480-3486.

[22] Ge S S, Cui Y J. Dynamic motion planning for mobile robots using potential field method[J]. Autonomous robots, 2002, 13(3): 207-222.

[23] Huang L. Velocity planning for a mobile robot to track a moving target—a potential field approach[J]. Robotics and Autonomous Systems, 2009, 57(1): 55-63.

[24] Willms A R, Yang S X. Real-time robot path planning via a distance-propagating dynamic system with obstacle clearance[J]. IEEE Transactions on Systems, Man, and Cybernetics, Part B (Cybernetics), 2008, 38(3): 884-893.

[25] Zhou J, He R, Wang Y, et al. Autonomous Driving Trajectory Optimization with Dual-Loop Iterative Anchoring Path Smoothing and Piecewise-Jerk Speed Optimization[J]. IEEE Robotics and Automation Letters, 2020.

[26] Stellato B, Banjac G, Goulart P, et al. OSQP: An operator splitting solver for quadratic programs[J]. Mathematical Programming Computation, 2020: 1-36.

[27] Faragher R. Understanding the basis of the kalman filter via a simple and intuitive derivation [lecture notes][J]. IEEE Signal processing magazine, 2012, 29(5): 128-132.

[28] Ang K H, Chong G, Li Y. PID control system analysis, design, and technology[J]. IEEE transactions on control systems technology, 2005, 13(4): 559-576.

[29] Dolgov D, Thrun S, Montemerlo M, et al. Practical search techniques in path planning for autonomous driving[J]. Ann Arbor, 2008, 1001(48105): 18-80.

[30] LaValle S M. Rapidly-exploring random trees: A new tool for path planning[J]. 1998.

致 谢

首先，感谢我的指导老师莫老师，他在整个毕设期间多次为我指点迷津，给予了许多帮助和支持。感谢我的博士导师何老师，他为我指出毕设的研究方向，并始终关心着我的研究进展。感谢实验室的师兄们，嘉伦师兄不厌其烦地解答我的疑问，帮我找资料、想方法，带着我一步步完成了算法主体流程的设计；祥宇师兄规范了我的论文写作，提出了许多有价值的修改意见。

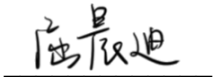
其次，衷心感谢我的家人们，感谢父母二十多年的养育之恩，没有你们的支持和鼓励我无法成长为现在的我，不论是分享喜悦还是遭遇困难挫折，你们总是最好的倾听者和帮助者，希望你们可以一直健康、快乐。想念去年离开了我的爷爷奶奶，多希望我还能成为你们的骄傲，孙女永远怀念你们。

此外，感谢我本科期间的老师、同学们，感谢清华园，让我度过了充实、精彩的四年。感谢我全天下最好的室友。感谢陪伴我十年之久的两位好朋友。感谢老e，他的直播陪我度过了许多做毕设的夜晚；感谢 Tom Hiddleston, Benedict Cumberbatch 等，作为我的偶像和人生导师。

最后，感谢自己，以我最喜欢的一句话作为论文的结尾，“世界上只有一种英雄主义，就是在看清生活的真相后依然热爱生活”。

声 明

本人郑重声明：所呈交的学位论文，是本人在导师指导下，独立进行研究工作所取得的成果。尽我所知，除文中已经注明引用的内容外，本学位论文的研究成果不包含任何他人享有著作权的内容。对本论文所涉及的研究工作做出贡献的其他个人和集体，均已在文中以明确方式标明。

签　名：  　　日　期：2021.5.27

附录 A 外文资料的书面翻译

使用四旋翼飞机在杂乱环境中跟踪运动目标

摘要——我们解决了使用四旋翼飞机在杂乱环境中跟踪运动目标这一具有挑战性的问题。我们的在线轨迹规划方法能够生成平滑、动态可行且无碰撞的多项式轨迹跟随视觉追踪的移动目标。当观察到目标时，可以估计目标轨迹并预测短时间范围内的目标运动。我们提出了一种将有限水平跟踪误差和四旋翼控制成本嵌入到二次规划（QP）的成本函数中的公式，同时将避免碰撞和动态可行性都编码为QP的线性不等式约束。我们生成跟踪轨迹的方法在毫秒级，因此适用于感测范围有限的在线目标跟踪。我们在配备了摄像头，激光测距仪，IMU和机载计算的四旋翼试验台上实施我们的方法。进行统计分析，模拟和实际实验以证明该方法的有效性。

I. 简介

自主航空机器人的估算[1]，控制[2]和计划[3]技术的成熟度使其能够在复杂的室内和室外3D环境中实现自主，从而为航空摄影，监控，检查，搜索和救援等领域的应用打开了广阔的市场。这些应用中的许多应用都需要空中机器人来自动跟随移动的目标。但是，在杂乱的环境中如何避免碰撞跟踪/跟踪自主目标的问题仍然存在，这是由于问题中混合了多个可能相互冲突的约束条件。例如，由于空中机器人具有惯性，与目标相比，空中机器人的控制权较低，以及在不同的飞行高度上有不同的障碍物结构，对目标的“僵硬”跟踪可能会导致在跟随时碰撞障碍物，同时目标的快速转向或停止可能会导致跟踪轨迹对于空中机器人而言是实际不可行的。我们检查在杂乱环境中目标跟踪的权衡和约束，不希望进行幼稚的“僵硬”跟踪。对于视觉跟踪而言尤其如此，在这种情况下，只要大部分时间目标都保持在视觉传感器的视场内，那么空中机器人与目标之间的距离的微小变化甚至是偶尔的遮挡都是可以接受的。但是，存在一些严格的约束条件，例如避障和动态可行性，必须予以满足，否则系统将发生崩溃。我们强调了可行性问题，因为生成的轨迹永远都不需要四旋翼超过其最大速度和加速度曲线。否则，四旋翼飞行器可能无法遵循该轨迹，从而最终导

致碰撞。因此，我们愿意降低跟踪质量，以确保议案的安全性和可行性。我们提出了一种公式，可将整个跟踪范围内的跟踪误差以及四旋翼控制成本嵌入到二次规划（QP）的成本函数中，同时将避免碰撞和平台动力学均编码为QP的线性不等式约束。已经提出了有希望的结果来解决使用车载传感器估计和预测目标运动的问题。使用RGB-D摄像机，[4]提出了FollowMe系统，该系统可以从背景中分割目标，然后生成用于跟踪飞行的航点。利用先进的计算机视觉技术，[5] - [8]提出了仅使用单眼相机的估计系统来计算相对于空中平台的目标运动。特别地，[8]中的方法能够在已知目标高度时精确地重建目标轨迹。 [9]提出了一种方法，其中目标位置在道路上约束场景被建模为多峰高斯分布，并使用贝叶斯估计进行更新。即使目标不在传感器范围内，四旋翼也可以采取相应的宏动作。但是，在这项工作中，我们专注于轨迹规划和跟踪控制问题，并使用人工标记获得3D目标相对于空中机器人的相对位置。在先前的文献中，轨迹规划和目标跟踪的控制问题通常在本地控制设置中处理。由于相机已成为各种机器人的标准设备，因此许多最近的工作都采用了基于视觉的控制方法[10]。[11] - [13]开发了基于视觉的控制方法，使四旋翼飞机可以利用在线视频流来跟踪选定的运动地面目标。但是，由于他们不考虑飞行空间中是否存在障碍物，因此他们的方法仅限于在广阔空旷地区的少数应用。在[14]中提出了一种简单的基于势场的方法，该方法由外部运动捕获系统辅助。为了确保飞行安全，Jeremy H. Gillula等人引入了通过可达性保证安全的在线学习（GSOLR）[15，16]，其中四旋翼通过计算状态空间中的保留集来避免选取会导致碰撞的控制信号。通过最小化具有安全约束的条件熵来选择控制信号。然而，除了一些简单的障碍，该方法的计算成本会随着障碍的复杂变得很大。它也无法考虑全局信息，容易陷入局部最小值中。这项工作的关键贡献是一种算法，该算法可生成平滑的轨迹用于目标跟踪，并确保安全性和动力学可行性。我们将轨迹公式化为样条曲线（分段多项式），如[3]中所述，这要归功于四旋翼的微分平坦性。通过从我们以前的方法[17，18]的开发中可以确保避免碰撞和动态可行性，该方法侧重于固定目的地的轨迹规划。从对目标的嘈杂视觉观察开始，我们使用多项式拟合来估计和预测目标运动。然后，我们将整个预测范围的误差与四旋翼控制成本（加加速度的积分）一起公式化为成本函数，以在QP中将其最小化。通过在线构建的全球3D地图，避免碰撞和动力学可行性被描述为QP的线性不等式约束。这样，我们将跟踪误差编码为整个时间范围内的“软约束”，同时确保安全性为“硬约束”。与基于局部控制的方法相比，我们基于

轨迹生成的方法结合了全局障碍信息，避免陷入局部最小值。换句话说，只要通过机载传感器获得障碍物的位置，四旋翼飞行器就可以找到复飞轨迹（如果存在的话），该复飞轨迹可以保证到达目标位置。多亏了预测的短视距目标运动，我们甚至能够在视觉测量的短期损失后恢复，如图7（a）和图8（a）所示。我们一口气优化了所有限制条件，并提供了参数调整界面，以适应不同平台之间的不同动力特性。我们将系统部署在配备了多个摄像头，激光测距仪，IMU和机载计算机的四旋翼试验台上。我们的在线实验表明，就我们所知，我们是第一个在未知，混乱的环境中实现对移动目标同时进行空中跟踪和避免碰撞的团队。

II. 概述

A. 问题表述

目标跟踪是一个复杂的工程系统，涉及硬件、本地化、映射、计划和控制中的技术模块。在这项工作中，我们专注于系统的轨迹规划部分。我们在传感和目标检测上使用工程解决方案[19]，在空中导航上使用最新技术成果[20]，以形成一个完整的空中跟踪系统。对于轨迹规划，我们考虑具有以下假设的设置：

1) 障碍物最初对于空中机器人是未知的，但是可以通过具有有限感测范围的机载全向范围传感器在线检测到。

2) 期望目标具有平稳的运动，并且速度和加速度会发生有限的变化。

3) 可以在线估算四旋翼状态。

4) 可以在线观察目标的嘈杂的相对姿势。

5) 预期四旋翼飞行器将保持相对于移动目标的相对位置$\mathbf{d}_s \in R^3$，并且对其方向进行单独控制以使其面向目标。

B. 系统架构

轨迹规划的算法流程如图2所示。为了有效地表示环境，障碍物信息存储在以八叉树结构组织的可变大小3D立方网格中[21]。利用对机载传感器提供的目标位置的观察，可以使用多项式拟合来估算目标轨迹，然后在有限的时间范围内对其进行预测（图3（b））。然后，我们将估计的目标轨迹沿$\mathbf{d}_s$方向移动，以获取跟踪/跟踪任务的“刚性”轨迹（图3（c））。之后，可以找到移动的目标轨迹经过的地图网格的集合。这些网格形成一个网格路径，称为初始飞行走廊（图3（d））。使用多起点A*方法调整与障碍物相交的初始飞行通道的部分，

以找到无碰撞的复飞通道（图3（f））。使用A*（图3（f））将此飞行走廊进一步与当前的四旋翼位置相连，然后进行扩充为四旋翼生成最终的无碰撞飞行走廊（图3（g））。最后，我们使用基于QP的方法来生成适合飞行走廊的轨迹，同时满足所有车辆动力学约束（图3（h））。在我们的实验中，整个管道以5 Hz的固定频率循环。

整个系统的架构如图2所示。所有没有标号的模块均来自我们先前关于状态估计，映射和控制的研究[20]。

III. 目标运动的估计和预测

我们表示目标在时间$t \in R$处的3D位置为$T(t) \in R^3$。由于假定目标运动是平滑的，因此我们可以使用泰勒展开式重写运动，并通过省略度数较高的项来近似运动：

$$\mathbf{T}(t) = \sum_{i=0}^{\infty} \frac{\mathbf{T}^{(i)}(0)}{i!} \times t^i, \tag{1}$$

$$= \sum_{i=0}^{n_t} \frac{\mathbf{T}^{(i)}(0)}{i!} \times t^i + O(t^{n_t}) \tag{2}$$

$$\approx \begin{bmatrix} \frac{\mathbf{T}^{(0)}(0)}{0!} \\ \frac{\mathbf{T}^{(1)}(0)}{1!} \\ \vdots \\ \frac{\mathbf{T}^{(n_t)}(0)}{n_t!} \end{bmatrix}^T \begin{bmatrix} 1 \\ t \\ \vdots \\ t^{n_t} \end{bmatrix} \tag{3}$$

$$= \hat{\mathbf{T}}(t), \tag{4}$$

然后，近似目标轨迹$\hat{T}(t)$变为关于t的n_t次多项式。

假设可以使用机载传感器获得目标相对于四旋翼的位置，我们可以使用四旋翼的位置对其进行转换，以获得全局框架中一系列嘈杂的目标位置观测值。令t0为当前时间，并且在时间段[t1，t0]中有L个观测值，其中t1是滑动窗口的开始时间。通过最小化所有L个观测值的距离误差，使用经典回归学习多项式（III）：

$$\min_{\hat{\mathbf{T}}(\cdot)} \sum_{i=0}^{L} \|\hat{\mathbf{T}}(t_i) - \mathbf{p}_i\|_2^2, \tag{5}$$

其中pi是在时间ti处观察到的目标3D位置。为了避免过拟合，我们添加加速度调节器，并结合现有的认知，即目标运动状态不会急剧变化：

$$\min_{\hat{\mathbf{T}}(\cdot)} \underbrace{\sum_{i=0}^{L} \|\hat{\mathbf{T}}(t_i) - \mathbf{p}_i\|_2^2}_{\text{estimation residual}} + L\lambda_t \underbrace{\int_{t_l}^{t_m} \|\hat{\mathbf{T}}^{(2)}(t)\|_2^2 dt}_{\text{acceleration regulator}}, \quad (6)$$

其中λt用于调整调节器的权重，t_m是预测时间上限。注意到[t_l，t_0]是我们实际观测移动目标的时间段，[t_0，t_m]是我们很短的一段预测时间，该预测用于跟踪的生成轨迹（第五节）。由于l^2范数的整数多项式具有二次封闭形式，（6）变为一个无约束的具有$\hat{T}(\cdot)$封闭形式解[22]的QP问题。

IV. 基于八叉树的环境表示

如我们先前的工作[18]一样，我们使用八叉树数据结构[23]将3D环境离散化为一个大小可变的3D立方网格集合。在此数据结构中，我们最初使用大网格来表示整个空间。如果网格中存在障碍物，我们将其划分为八个子网格，并标记充满障碍物的网格。我们将该策略递归应用于每个子网格，直到该子网格没有障碍或达到最佳分辨率为止。从仅使用几个大网格表示大量可用空间的意义上来说，此数据结构可提高内存效率。对于容量占用查询，它还具有合适的对数运行时复杂度，如[18]所示。

此外，为了使用基于搜索的方法以在线方式查找有效的飞行走廊，在线维护表示网格之间的连通性信息的图形[18]。当观察到障碍物时，此图将在本地更新。这样可以节省大量在线计算资源，并且在理论上和实践上都证明是有效的。

V. 目标跟踪的轨迹生成

在本节中，我们将详细介绍用于给定预测的目标运动，确定无碰撞跟踪轨迹。过程展示在了图3（c）-3（h），带有节号的标题便于可视化我们的方法。

A. 飞行走廊的产生

在跟踪任务期间，四旋翼应该跟目标保持一个固定的距离$\mathbf{d}_s \in R^3$。在最理想的情况下，即没有障碍物和动力学约束，四旋翼应跟随轨迹$\mathbf{T}_s(t) = T(t) + d_s$，$\mathbf{T}_s(t)$也是一个多项式，其形式$\mathbf{T}_s(t) = \sum_{j=0}^{n_t}[c_j \cdot t^j]$，其中$\{\mathbf{c}_j \in R^3 | 0 \le j \le n_t\}$是系数。

基于八叉树的地图使我们能够快速找到$\mathbf{T}_s(t)$穿过的一系列连接网格，称为飞行走廊。其中一些网格可能会被障碍（图3（d））。因此，我们执行一次多点启动A*搜索以找到一系列复绕网格（图3（f））。可以通过将阻塞网格的前

一个网格设置为目的地，并将所有可用的后续网格设置为起点来完成此操作。启发式函数是到目标的欧几里得距离。该复飞走廊的起始网格进一步与当前的四旋翼位置相连，以形成具有m个网格的无碰撞飞行走廊（图3（e））。该搜索将提供一条路径，该路径使四旋翼飞机尽可能快地靠近偏移的目标轨迹$T_s(\cdot)$，同时避免障碍物。然后，如[18]所示，这个无碰撞的飞行走廊将被扩充成m个大的重叠网格，以扩大我们基于优化的轨迹生成的配置空间（图3（g））。

B. 微分平坦度和多项式轨迹

如[3]所示，四旋翼具有差动平坦性。这使我们能够根据简单的平面输出（例如x，y，z位置和偏航角以及它们的导数）来表达四旋翼飞行器的完整状态。在这项工作中，我们对偏航角使用简单的PD控制，以使车载摄像头始终面向目标。对于3D位置，我们利用平面度属性并将四旋翼轨迹表示为三个单独的分段多项式。对于每个维度，我们都有：

$$\mathbf{f}(t)=\begin{cases}\sum_{j=0}^{n}[\mathbf{a}_{1j}\cdot(t-t_0)^j] & t_0\le t\le t_1\\ \vdots & \\ \sum_{j=0}^{n}[\mathbf{a}_{ij}\cdot(t-t_{i-1})^j] & t_{i-1}\le t\le t_i\\ \vdots & \\ \sum_{j=0}^{n}[\mathbf{a}_{Mj}\cdot(t-t_{m-1})^j] & t_{m-1}\le t\le t_m\end{cases},\quad(7)$$

轨迹的每段都是在指定的时间间隔$[t_{i-1},t_i]$上的n^{th}次多项式。$a_{ij}\in R^3$是i^{th}段的j^{th}多项式系数。我们的多项式段数等于飞行走廊中的网格数（m）。

C. 基于优化的轨迹生成

我们现在介绍一种新颖的基于QP的方法来生成多项式，如图3（h）所示，该多项式遵循移动的目标轨迹（图3（c）），同时仍完全适合飞行通道，并避免碰撞。

理想情况下，如图3（c）所示，移动后的目标轨迹$T_s(\cdot)$应该跟随四旋翼以达到最佳跟踪效果。但是，由于$T_s(\cdot)$可能与障碍物相交，因此通常不可能使用$T_s(\cdot)$作为实际飞行轨迹。而且，目标（例如人）的运动能力可能优于具有突然停止和急转弯的四旋翼飞机的运动能力，因此四旋翼飞机可能无法精确地跟随目标。最好权衡一下跟踪精度以获得平稳的飞行，并使用适合四旋翼飞行器动态能力的控制命令。为此，我们选择最小化受两个因素影响的成本函数：1）在目标运动整个预测范围内的跟踪误差； 2）在整个跟踪范围内的加加速度积分。第一项使四旋翼飞机接近预测的目标轨迹，而第二项将跟踪的控制消耗最小化。四旋翼的动态能力被视为线性不等式约束，以约束平台的轨迹加速度和加

速度。我们将避免碰撞转化为走廊边界约束，可以表示为线性不等式。这足够使适合飞行走廊的轨迹是无碰撞的。膨胀的飞行走廊可以覆盖大量的空间，从而为四旋翼飞机提供了灵活的动力用于目标跟踪的安全飞行区域。基于优化的轨迹生成的完整公式如下：

$$\min_{\mathbf{f}(\cdot)} \underbrace{\int_{t_0}^{t_m} \|\mathbf{f}(t)-\mathbf{T}_s(t)\|_2^2 dt}_{\text{tracking error}} + \lambda \underbrace{\int_{t_0}^{t_m} \|\mathbf{f}^{(3)}(t)\|_2^2 dt}_{\text{jerk regulator}} \tag{8}$$

$$\text{s.t.} \quad \underbrace{\mathbf{f}^{(j)}(t_0)=\mathbf{f}_0^j}_{\text{initial state constraint},\forall j\in\{0,1,2\}} \tag{9}$$

$$\underbrace{\mathbf{f}^{(j)}(t_m)=0}_{\text{stopping policy constraint},\forall j=0\ldots2} \tag{10}$$

$$\underbrace{\lim_{t\to t_i-}\mathbf{f}^{(j)}(t)=\lim_{t\to t_i+}\mathbf{f}^{(j)}(t)}_{\text{smoothness constraint},\forall t=2\ldots m,j=0\ldots2} \tag{11}$$

$$\underbrace{\mathbf{f}(t)\in\Omega_p(t)}_{\text{safety constraint}} \tag{12}$$

$$\underbrace{\mathbf{f}^{(1)}(t)\in\Omega_v(t)}_{\text{velocity constraint}} \tag{13}$$

$$\underbrace{\mathbf{f}^{(2)}(t)\in\Omega_a(t)}_{\text{acceleration constraint}}, \tag{14}$$

其中$\Omega\,\mathrm{p}(\mathrm{t})$，$\Omega\,\mathrm{v}(\mathrm{t})$和$\Omega\,\mathrm{a}(\mathrm{t})$分别表示在时间t处的飞行通道、可达到的速度和可达到的加速度。 λ是用于调整跟随“刚度”的加权项。较小的λ会产生轨迹紧密跟踪目标运动，而较大的λ通过降低跟踪性能来平滑轨迹。

对于第i^{th}段，移动目标轨迹$\mathbf{T}_s(\cdot)$和跟踪轨迹$\mathbf{f}(\mathrm{t})$都是多项式，并且（8）中的跟踪误差项表示为目标与无人机沿整个轨迹的欧氏距离平方的积分。选择正方形形式不仅因为它在域内是平滑的，还因为它允许闭合形式的积分解。在时间间隔$[\mathrm{t}_{\mathrm{i}-1}, t_i]$内，对应于第$\mathrm{i}^{\mathrm{th}}$段，我们将移动的目标轨迹$\mathbf{T}_s(\cdot)$重写为：

$$\mathbf{T}_s(t)=\underbrace{\begin{bmatrix}1\\ t-t_{i-1}\\ \vdots\\ (t-t_{i-1})^n\end{bmatrix}^T}_{\mathbf{t}_i}\underbrace{\begin{bmatrix}\sum_{k=0}^{n_t}\left[\binom{k}{0}\cdot t_i^{k-0}\cdot\mathbf{c}_{ik}^T\right]\\ \sum_{k=1}^{n_t}\left[\binom{k}{1}\cdot t_i^{k-1}\cdot\mathbf{c}_{ik}^T\right]\\ \vdots\\ \sum_{k=n}^{n_t}\left[\binom{k}{n}\cdot t_i^{k-n}\cdot\mathbf{c}_{ik}^T\right]\end{bmatrix}}_{C_i}=\mathbf{t}_i{}^T\mathbf{C}_i \tag{15}$$

同样地，我们可以将i^{th}段的跟随轨迹重写为：

$$\mathbf{f}(t) = \underbrace{\begin{bmatrix} 1 \\ t - t_i \\ \vdots \\ (t - t_{i-1})^n \end{bmatrix}^T}_{\mathbf{t}_{i-1}} \underbrace{\begin{bmatrix} \mathbf{a}_{i0}^T \\ \mathbf{a}_{i1}^T \\ \vdots \\ \mathbf{a}_{in}^T \end{bmatrix}}_{\mathbf{A}_i} = \mathbf{t}_i{}^T \mathbf{A}_i, \tag{16}$$

现在移动目标轨迹和追踪轨迹都是相同形式了，将维度抽象为$l \in \{x, y, z\}$，我们可以把跟踪误差的积分改写成：

$$\int_{t_{i-1}}^{t_i} \|\mathbf{f}(t) - \mathbf{T}_s(t)\|_2^2 dt \tag{17}$$

$$= \int_{t_{i-1}}^{t_i} \sum_{l \in \{x,y,z\}} \left[\mathbf{t}_i^T (\mathbf{A}_i^l - \mathbf{C}_i^l)\right]^T \left[\mathbf{t}_i^T (\mathbf{A}_i^l - \mathbf{C}_i^l)\right] dt \tag{18}$$

$$= \sum_{l \in \{x,y,z\}} (\mathbf{A}_i^l - \mathbf{C}_i^l)^T \underbrace{\int_{t_{i-1}}^{t_i} (\mathbf{t}_i \mathbf{t}_i^T) dt}_{\mathbf{Q}_i} (\mathbf{A}_i^l - \mathbf{C}_i^l) \tag{19}$$

$$= \sum_{l \in \{x,y,z\}} \mathbf{A}_i^{l\,T} \mathbf{Q}_i \mathbf{A}_i^l - 2\mathbf{b}_i^{l\,T} \mathbf{A}_i^l + \mathbf{C}_i^{l\,T} \mathbf{Q}_i \mathbf{C}_i^l, \tag{20}$$

其中$\mathbf{b}_i^l = Q_i^T C_i^l$和$\mathbf{Q}_i$的定义见公式（19）。同样地，$i^{th}$段的加加速度积分项也可以改写成二次形式：

$$\int_{t_{i-1}}^{t_i} \|\mathbf{f}^{(3)}(t)\|_2^2 dt \tag{21}$$

$$= \int_{t_{i-1}}^{t_i} \left\| \underbrace{\begin{bmatrix} 0 \\ 0 \\ 0 \\ \frac{3!}{0!} \cdot (t - t_{i-1})^0 \\ \frac{4!}{1!} \cdot (t - t_{i-1})^1 \\ \vdots \\ \frac{n!}{(n-3)!} \cdot (t - t_{i-1})^{n-3} \end{bmatrix}^T}_{\mathbf{t}_i^r} \underbrace{\begin{bmatrix} \mathbf{a}_{i1}^T \\ \mathbf{a}_{i2}^T \\ \mathbf{a}_{i3}^T \\ \mathbf{a}_{i4}^T \\ \mathbf{a}_{i5}^T \\ \vdots \\ \mathbf{a}_{in}^T \end{bmatrix}}_{\mathbf{A}_i} \right\|_2^2 dt \tag{22}$$

$$= \sum_{l \in \{x,y,z\}} \mathbf{A}_i^{l\,T} \underbrace{\int_{t_{i-1}}^{t_i} \mathbf{t}_i^{r\,T} \mathbf{t}_i^r dt}_{\mathbf{H}_i} \mathbf{A}_i^l \tag{23}$$

$$= \sum_{l \in \{x,y,z\}} \mathbf{A}_i^{l\,T} \mathbf{H}_i \mathbf{A}_i^l, \tag{24}$$

其中$\mathbf{H}_i$在公式（23）中定义。$\mathbf{A}_i^l$和$\mathbf{C}_i^l$分别是$\mathbf{A}_i$和$\mathbf{C}_i$的第1行，$\mathbf{A}_i^l$和$\mathbf{C}_i^l$的定义在（15）（16）行。将跟随误差和正则化项结合起来，我们可以将（8）表示为QP的标准成本函数：

$$\min_{\mathbf{A}_i} \sum_{i=1}^{m} \sum_{l \in \{x,y,z\}} \left[\mathbf{A}_i^{l^T} (\mathbf{Q}_i + \lambda \mathbf{H}_i) \mathbf{A}_i^l - 2\mathbf{b}_i^{l^T} \mathbf{A}_i^l \right]. \quad (25)$$

由于f(·)是分段多项式函数，因此可以将初始状态（9），最终状态（10）和平滑度要求（11）直接表示为线性相等约束。为了在整个跟踪轨迹上施加安全性（12），速度（13）和加速度（14）约束，我们利用了先前工作中提出的经过验证的方法[17，18]。我们首先生成跟踪轨迹，而不考虑上面的任何间隔约束。由于安全性，速度和加速度约束实际上是多项式的0、1、2阶导数的区间边界，因此我们可以检查生成的多项式及其导数的所有极值点，并添加违反道路边界或超出预设的速度/加速度极限，这是对QP的点约束。我们迭代执行此过程，直到整个轨迹中不存在任何约束冲突。我们的方法已经证明了需要添加到QP的约束点数量的理论界限。还值得指出的是，实际方案所需的约束点的实际数量通常远小于理论最大值。事实上，正如仿真和真实实验（第VI节）所证明的那样，我们的方法在实践中非常快。我们建议读者参考[17，18]了解QP约束的详细公式以及我们方法的理论证明。

通过上述公式，可以通过解决标准QP问题来完成跟踪轨迹的生成。此外，在我们的案例中，QP公式稀疏，这表明可以使用开源QP求解器进一步优化计算时间。

VI. 实验结果

我们进行了两次仿真和一次真实的飞行实验，以证明所提出方法的性能。为了实现实时操作，我们使用带有-O3优化选项的g++编译器在C++11中实现该系统。流行的开源线性代数库Eigen和开源二次规划求解器，面向对象的二次编程软件（OOQP）[24]，用于构造和求解轨迹生成的QP。在所有测试案例中，四旋翼飞机起飞时都没有任何有关环境的先验知识，整个管道以5Hz的固定频率循环，以实现实时障碍物检测和轨迹重新规划。

A. 调整跟踪的“刚度”

在此仿真中，我们显示了跟踪调节器λ对整体跟踪性能的影响，如Sect V-C中的（8）所定义的那样。我们将λ的不同值设置为0.001至10。如图5和6所示，较小的λ允许进行更多的敏捷运动，从而导致“较陡峭”的跟踪轨迹和较小的跟踪误差。另一方面，较大的λ会使轨迹更平滑，速度分布的变化较小，因此对于动力学较慢的平台更容易跟随。如预期的那样，跟踪误差随着敏捷轨迹的减少而增加。可以看出，目标运动通常比跟踪平台更敏捷。但是，我们能

够在所有λ值的情况下保持合理的跟踪性能。λ提供了一种方便的方法来调整跟踪性能，以使跟踪轨迹的敏捷性水平与用户的期望相匹配。例如，我们可能需要”stiffer”跟踪来监视重点放在哪里的任务在目标本身上。另一方面，我们可能会选择对航拍进行“更柔和”的跟踪，重点是周围的场景。不过，需要强调的是，无论我们如何调整λ，对于飞行平台而言，所产生的跟踪轨迹始终是安全且动态可行的，因为这些限制是在我们的QP公式中强制执行的（第V-C节）。

B. 模拟目标跟踪

在本节中，我们介绍了四旋翼飞机在最初未知且混乱的环境中跟踪运动目标的仿真结果。我们将目标检测传感器的范围设置为10 m，视野为90度。目标观测值被零平均值和0.1 m的标准偏差的高斯噪声所破坏。用模拟激光测距仪在10 m的范围内观察障碍物。

在图7（a）中，我们以黄色突出显示跟踪轨迹，以红色突出显示目标运动。当目标突然改变速度（A，B，E）时，四旋翼会过冲一点。然后，需要一点点复飞来避开障碍物（C），并由于新的障碍物（D）找不到动态可行的复飞轨迹，因此紧急停止。当避免障碍物突然进入感应范围时，模拟四旋翼的速度会迅速变化（图7（b））。尽管四旋翼主动地降低了跟踪性能以换取飞行安全，但从图7（c）中我们可以看到跟踪误差总是有界的。我们还显示了系统每个组件的运行时性能。整个流水线的最大计算时间约为60ms，这表明具有实时功能（图7（d））。

C. 真实世界的实验

现在，我们提出在未知，混乱的室内环境中跟踪移动地面车辆的四旋翼飞机的真实实验。四旋翼试验台基于如图1所示的DJI Matrice 100平台，并配备了Intel NUC（IntelCore i5425OU，16GB RAM），镜头朝下的光流摄像头，用于目标检测的45度倾斜摄像头以及2D北洋激光测距仪用于定位和障碍物检测。地面车辆带有人工视觉标记，易于检测。实验的快照和可视化分别可以在图1和图8（a）中找到。如图8（b）所示，目标观测结果非常嘈杂。然而，由于目标运动的多项式拟合和平滑的轨迹生成算法，四旋翼仍然能够以合理的方式跟随目标。我们着重指出，在实验过程中，地面车辆进入了一个洞在墙（图1（c））上，会导致临时的跟踪损耗，这在图8（c）中也得到了证明。但是，四旋翼仍然能够使用预测的目标运动来完成回避动作，并在壁的另一侧重新观察目标。运行时间统计数据如图8（d）所示，再次证明了实时能力。

VII. 结论与未来工作

在本文中，我们提出了一种新颖的轨迹规划方法，用于四旋翼无人机追踪移动目标。该方法结合全新的QP公式和先前引入的飞行走廊概念，将跟踪误差最小化并将安全性和动力学可行性的保证结合到可以在线解决的单个优化问题中。据我们所知，我们的真实世界实验是第一个演示了在未知，混乱环境中进行实时目标跟踪并具有防撞和动力学可行性保证的实验。在我们研究的下一阶段，旨在将基于视觉的目标检测方法集成到我们的系统中，以实现无标记目标跟踪。

书面翻译对应的原文索引

[1] J. Chen, T. Liu and S. Shen, "Tracking a moving target in cluttered environments using a quadrotor," 2016 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS), 2016, pp. 446-453, doi: 10.1109/IROS.2016.7759092.

综合论文训练记录表

学生姓名	屈晨迪	学号	2017010928	班级	自 71
论文题目	基于数据驱动的移动目标预测与拦截				
主要内容以及进度安排	主要内容：本文设计了一套算法流程，旨在解决环境中存在静态或动态障碍物时，具有不定运动规律的移动目标的拦截问题。算法首先对移动目标进行观测，基于观测得到的历史轨迹数据，通过多项式拟合的方法在线预测目标的未来轨迹，接着利用目标函数辅助选取拦截点，最后采用路径和速度解耦的方式进行轨迹规划。同时自主搭建了仿真环境，在仿真小车上部署并测试了本算法，成功实现了移动目标在匀速直线、曲线等运模式下的拦截。 进度安排：2020 年 11 月论文开题，2020 年 12 月深入进行决策规划部分的文献调研，2021 年 1 月-2 月学习利用 ros 和 gazebo 搭建仿真平台，2-3 月设计了感知、预测和控制部分的算法并部署在仿真环境上，4 月初进行中期答辩，4 月-5 月初设计部署了决策规划部分代码，并联合调试，5 月下旬撰写论文，准备答辩。 指导教师签字： 考核组组长签字： 2020 年 11 月 27 日				
中期考核意见	课题取得了一定进展，后期需要加大时间投入，确保完成计划的任务。 考核组组长签字： 2021年 4月 9 日				

指导教师评语	屈晨迪同学在毕设期间针对移动目标的拦截问题进行了调研和探究，设计了完整的拦截算法，通过观测目标、预测目标未来轨迹、拦截点决策和轨迹规划几步实现了智能体对移动目标的拦截，并利用 ROS 和 Gazebo 搭建仿真平台，部署测试了其算法。毕业设计过程中态度端正、积极主动，课题将理论与实验相结合，展现了自己的思考。整体工作量充实，达到了综合论文训练课程的要求。 指导教师签字：莫一林 2021 年 6 月 5 日
评阅教师评语	论文研究了移动目标的拦截问题，介绍了当前移动目标拦截问题的背景和研究现状，较好描述了课题中采用的算法思路和实现方案，并具体分析了仿真测试得到的结果。论文撰写较规范，语言表述较为流畅，同意参论文答辩。 评阅教师签字： 2021 年 6 月 3 日
答辩小组评语	屈晨迪同学在答辩过程中，阐述清楚，回答问题正确。答辩小组老师们一致认为该生的毕业论文达到了学士学位论文水平，同意答辩通过。 答辩小组组长签字： 2021 年 6 月 4 日

总成绩：B+

教学负责人签字：

2021 年 6 月 15 日