西南科技大学

Southwest University of Science and Technology

专业学位硕士研究生学位论文

（电子信息硕士）

视觉数据驱动的多旋翼无人机自主飞行关键技术研究

作者姓名 钟良涛

学　　号 7220200296

导　　师 楚红雨

答辩委员会成员 王坤朋（主席） 苏春晓

李华锋 郭玉英 白克强

评　阅　人 曾云南 朱静

2023 年 5 月

分类号：　　学校代码：10619
UDC：　　密级：

西南科技大学硕士学位论文

（电子信息硕士）

视觉数据驱动的多旋翼无人机自主飞行关键技术研究

钟良涛

学科(专业)或领域名称：电子信息
指导教师姓名、职称：楚红雨 教授
校外指导教师姓名、职称：杨小川 副研究员
学生所属培养单位：西南科技大学
论文答辩日期： 2023 年 5 月 25 日

A Thesis Submitted to Southwest University of Science and Technology for the Degree of Master

(Master of Electronic and Information Engineering)

Research on key technologies of vision driven autonomous flight for multi-rotor UAV

Candidate: Liangtao Zhong

Supervisor: Prof. Hongyu Chu

A.P. Xiaochuan Yang

May 2023

摘　要

多旋翼无人机由于其结构简单、机动性强、具备垂直起降和定点悬停能力等特点，近年来，在学术界和工业界得到了广泛研究和应用。由于可通过全球导航卫星系统（Global Navigation Satellite System, GNSS）获取较为准确的定位信息，多旋翼无人机在电力巡检、航拍摄影、快递配送等领域取得了较快进展。然而，对于 GNSS 受限的场景，多旋翼无人机自主飞行能力仍需进一步提升。近年来，随着机器视觉技术的快速发展，使得无人机利用低成本的视觉传感器在 GNSS 受限的场景进行自主飞行成为可能。本文则对视觉数据驱动的多旋翼无人机自主飞行关键技术展开研究，主要研究内容如下：

1. 针对视觉数据驱动的多旋翼无人机在复杂动态环境的定位问题，本文提出了一种基于静态特征点筛选的同时定位与建图算法。首先，利用特征点光流设计了描述运动模式的特征，并使用高斯混合模型对该特征进行聚类。然后，结合超轻量目标检测算法和光流特征聚类结果设计了静态特征点筛选算法，在过滤掉动态特征点的同时尽可能保留更多的静态特征点。最后，将其集成至 ORB-SLAM2 前端，并在 TUM RGBD 公开数据集上进行定量实验。结果表明，本文算法在高动态场景中的 ATE 均方根误差相比 ORB-SLAM2 平均降低96.0%，相比 DS-SLAM 平均降低46.2%，提高了 SLAM 在复杂动态环境的定位精度，且实时性满足无人机自主飞行的需要。

2. 针对视觉数据驱动的多旋翼无人机在复杂动态环境的运动规划问题，本文研究了基于样条曲线的多旋翼无人机局部运动规划算法。首先，使用深度相机构建局部栅格地图以表示障碍物环境。其次，使用样条曲线对无人机飞行轨迹进行参数化，并设计了轨迹与障碍物之间距离的估计算法。然后，基于障碍物距离设计多项罚函数，优化无人机在静态场景中的飞行轨迹。最后，设计了基于简易运动模型的动态障碍物规避方法使得无人机能够规避低速动态障碍物。实验表明，本文研究的局部运动规划算法在复杂环境能够生成平滑且安全的飞行轨迹。

3. 基于以上关键技术的研究成果，本文构建了视觉数据驱动的四旋翼无人机自主飞行系统。搭建了四旋翼无人机平台，完成了视觉传感器校准与标定，对本文算法进行了实现。在真实场景的实验结果表明，本文构建的系统能够在复杂环境下以$1.5m/s$安全自主飞行。

关键词：多旋翼无人机；自主飞行；视觉 SLAM；运动规划；视觉数据驱动
论文类型：应用研究

ABSTRACT

Multi-rotor UAV has been widely studied and applied in academia and industry in recent years due to its simple structure, strong maneuverability, vertical take-off and landing and fixed-point hovering ability. Due to the ability to obtain more accurate positioning information through Global Navigation Satellite System, multi-rotor UAV has made rapid progress in the fields of power inspection, aerial photography, and express delivery. However, for GNSS-limited scenarios, the autonomous flight capability of multi-rotor UAV still needs to be further improved. In recent years, with the rapid development of machine vision technology, it is possible for UAV to use low-cost vision sensors to fly autonomously in GNSS-limited scenes. In this thesis, the key technologies of autonomous flight for vision driven multi-rotor UAV are studied. The main contents are as follows:

(a) Aiming at the positioning problem for vision driven multi-rotor UAV in complex dynamic environment, this thesis proposes a simultaneous positioning and mapping algorithm based on static feature point screening. Firstly, the features describing the motion pattern are designed by using the optical flow of feature points, and the Gaussian mixture model is used to cluster the features. Then, combined with the ultra-lightweight target detection algorithm and the optical flow feature clustering results, a static feature point screening algorithm is designed to filter out dynamic feature points while retaining as many static feature points as possible. Finally, it is integrated into the ORB-SLAM2 front end and quantitative experiments are performed on the TUM RGBD public dataset. The results show that the ATE RMSE of the proposed algorithm in the high dynamic scene is 96.0% lower than that of ORB-SLAM2, and 46.2% lower than that of DS-SLAM. The positioning accuracy of SLAM in complex dynamic environment is improved, and the real-time performance meets the needs of UAV autonomous flight.

(b) Aiming at the motion planning problem for vision driven multi-rotor UAV in complex dynamic environment, this thesis studies the local motion planning algorithm of multi-rotor UAV based on spline curve. Firstly, a depth camera is used to construct a local grid map to represent the obstacle environment. Secondly, the spline curve is used to parameterize the flight trajectory of the UAV, and the estimation algorithm of the distance between the trajectory and the obstacle is designed. Then, a multiple penalty function is designed based on the obstacle distance to optimize the flight trajectory of the UAV in a static scene. Finally, a dynamic obstacle avoidance method based on a simple motion model is designed to enable the UAV to avoid low-speed dynamic obstacles. Experiments show that the local motion planning algorithm studied in this thesis can generate smooth and safe flight trajectories in complex environments.

(c) Based on the above key technical research results, this thesis constructs a vision driven four-rotor UAV autonomous flight system. The four-rotor UAV platform was built, the calibration and calibration of the visual sensor were completed, and the algorithm was implemented. The experimental results in real scenes show that the system designed in this

thesis can fly safely and autonomously at 1.5 m/s in complex environments.

KEY WORDS: Multi-rotor UAV; Autonomous flight; Visual SLAM; Motion planning; Vision driven.
TYPE OF THESIS: Application Research

目　录

1 绪论

1.1 研究背景和意义

无人驾驶飞行器，简称无人机，指利用远程遥控设备控制或利用机载传感器和飞控系统完成预定任务的不载人飞机[1]。无人机根据其气动布局可分为固定翼无人机、扑翼式无人机、多旋翼无人机和混合异构无人机等，由于多旋翼无人机结构简单、机动性强等特点在工业界和学术界得到了广泛的研究和应用[2–4]。随着无人机自主作业能力的提高，无人机在多种行业的应用已体现出显著的效率优势，如图 1-1 所示。

(a) DJI 电力巡检无人机解决方案

(b) Skydio 执法无人机

(c) 亿航快递配送解决方案

(d) Sony 摄影无人机

图 1-1 多旋翼无人机在各行业应用

在传统的电网精细巡检任务中，巡检人员要对电塔等设备细节进行故障排查，工作量大，专业化程度高，传统巡检方式难以获得结构化、标准化数据，为缺陷识别带来挑战。大疆创新（DJI）公司的电网精细化巡检无人机结合三维点云数据及航线，对设备进行高精度自动巡检，大幅提升巡检效率，如图 1-1(a)所示。Skydio 公司研发的执法无人机可以在 GNSS 拒止环境中利用计算机视觉进行定位，探索存在安全隐患的未知空间并对环境进行三维重建，保障执法人员的生命安全，如图 1-1(b)所示。亿航公司研发的快递配送无人机能够实现飞行器的自动充电、装载仓的自动装卸、包裹的自动分拣、包裹的自主寄取等，极大程度的提高了快递的配送效率，如图 1-1(c)所

示。Sony 公司研发的 Airpeak 无人机在保证无人机飞行安全的同时实现高速跟踪空中拍摄，对传统的升降机摄影方式带来了巨大的革新，如图 1-1(d)所示。随着无人机自主飞行能力的提高，多旋翼无人机在各行业中应用还存在巨大潜力。

从目前商用多旋翼无人机的应用场景来看，大部分无人机的作业范围局限于高空或空旷场景，使得无人机不需要考虑过多的避障问题，只需要规划出安全的离散的无人机航迹，利用由 GNSS 和 IMU 构成的组合导航系统即可进行自动化作业。而多旋翼无人机在 GNSS 受限的复杂环境下自主飞行作业还有进一步拓展的空间，例如利用多旋翼无人机进行隧道巡检、室内探测等。主要原因为多旋翼无人机在复杂环境下自主作业时需要对稠密或是动态的障碍物进行感知，并进行安全的运动规划，而目前的多旋翼无人机受限于机载计算能力、GNSS 受限[5]复杂环境下定位精度以及复杂环境运动规划难度较大等问题，在此类复杂场景下的无人机自主飞行技术仍具有一定挑战。

要提高多旋翼无人机在 GNSS 受限复杂场景下的自主作业能力，可利用机器视觉技术对 GNSS 进行补充，研究视觉数据驱动的自主飞行技术，但其中的关键技术还需要进一步突破。视觉数据驱动的无人机在复杂环境下安全自主飞行离不开同时定位与建图（Simultaneous Localization and Mapping, SLAM）[6–9]和运动规划（Motion Planning）[10–12]。现阶段众多 SLAM 算法在结构化场景中取得了不错的效果，而应用于复杂环境下无人机自主飞行的 SLAM 算法需要考虑无人机的数据处理能力和场景针对性的优化方法。运动规划算法则需要具有快速求解能力并满足无人机动力学约束以实现无人机在复杂障碍物环境中的安全轨迹规划。

以上分析可以发现，多旋翼无人机的自主飞行具有重要的研究意义，自主飞行所依赖的复杂环境下 SLAM 算法以及运动规划算法还有进一步研究的空间。本文则对视觉数据驱动的多旋翼无人机在复杂环境下自主飞行的其中两项关键技术（视觉 SLAM 与运动规划）展开研究。针对多旋翼无人机在复杂动态环境下的定位问题，提出一种基于静态特征点筛选的 RGBD SLAM 方法。针对多旋翼无人机在复杂障碍物环境下的运动规划问题，研究了基于样条曲线的多旋翼无人机局部运动规划算法。结合以上算法，本文还构建了视觉数据驱动的四旋翼无人机自主飞行系统，在仅依靠机载视觉传感器和机载有限算力的条件下，实现了无人机在复杂动态场景中的安全自主飞行。

1.2 国内外研究现状

随着科学研究的进一步深入和各行业需求的增加，无人机自主飞行技术得到了快速发展。在视觉数据驱动的同步定位与建图和运动规划方面国内外研究机构都做了大量研究工作，无人机自主飞行的其中两大关键技术已不再是停留在理论研究阶段。自主飞行技术发展历程中的思想和方法值得相关研究人员学习和借鉴。本文按照时间顺序梳理了无人机自主飞行的其中两大关键技术——视觉数据驱动的 SLAM 算法和运动规划算法的国内外研究现状，并在每个部分结尾给出了针对现状的结论和展望。

1.2.1 视觉数据驱动的 SLAM 研究现状

视觉数据驱动的 SLAM 算法指利用单目（Monocular）、双目（stereo）、RGBD 相机在内的视觉传感器进行定位和环境感知的方法[13]，在本文中简称为视觉 SLAM。

（1） 静态环境视觉 SLAM 研究现状

早在 2007 年，Davidson 和 Reid 等人[14]开创性的提出了基于单个相机的视觉 SLAM 算法，MonoSLAM。该方法使用传统基于滤波的方法同时估计相机位姿和环境地图。由于需要把每个特征信息加入 EKF 中进行优化，因此在大场景中的实时性将会有所降低。同年，Klein 等人[15]为了提升 MonoSLAM 的性能，提出了 PTAM（Parallel Tracking And Mapping），该算法首次将位姿估计和建图两个任务分配到 CPU 的两个线程，该方法不仅没牺牲算法的定位精度还提高了计算速度。此外，Klein 等人还在 PATM 算法中引入了关键帧、局部 BA、全局 BA 的概念，为后续视觉 SLAM 算法奠定了坚实的理论基础。2015 年，Mur-Arta 等人[16]进一步改进了 PTAM 算法，提出了 ORB-SLAM，该算法引入了回环检测线程，目的是利用机器人识别曾经到达的场景以优化轨迹和地图的累计误差。由于 ORB 特征对图像平移、旋转和光照变化鲁棒，ORB-SLAM 在跟踪、建图、回环检测、重定位时仅使用 ORB 特征以简化系统和提升稳定性。ORB-SLAM 使用可以描述场景相似度的词袋模型（Bag of Word）进行回环检测，然后执行全局 BA 以实现定位和建图的全局一致性。2017 年，Mur-Arta 等人[17]提出的 ORB-SLAM2 在第一代基础上进行了扩展，支持更多类型的传感器，包括单双目针孔相机和深度相机，同时 ORB-SLAM2 的地图可以离线保存，在第二次运行时能够进行重定位。ORB-SLAM2 也采用多线程方式并行计算，首先在跟踪线程使用重投影误差优化相机位姿，然后在局部建图线程对共视图关键帧使用 BA 构建局部地图，最后在回环处理线程中使用位姿图优化减小位姿和地图的累计误差。多个线程的协作使得 ORB-SLAM2 具有良好的扩展性和位姿估计精度，以至于现阶段仍有许多算法由 ORB-SLAM2 改进而来[18–21]。2018 年，香港科技大学秦通等人[22]提出了 VINS-Mono，该算法以特征点光流追踪和每帧图像间预积分开始，然后联合视觉传感器与 IMU 进行联合初始化，最后将视觉数据与 IMU 数据联合图优化获得精确的位姿。后来，秦通等人进一步扩展了 VINS-Mono，使其能够支持“双目”和“双目+IMU”，称之为 VINS-Fusion[23,24]。2020 年，Campos 等人[20]开源了 ORB-SLAM3，该算法支持单目、双目和 RGBD 相机与 IMU 联合进行位姿估计或仅利用视觉传感器进行位姿估计，同时视觉传感器还支持传统的针孔模型相机和具有更广视野范围的鱼眼模型相机。该算法对 ORB-SLAM2 进行了多处改进，使之比 ORB-SLAM2 定位精度准确 2~5 倍。2021 年，Yang 等人[25]提出了一种异步多相机 SLAM 算法，并将其应用于大范围室外场景机器人定位。该算法使用多个视场重叠范围较小的相机扩大视野范围，提高了算法在复杂环境下的鲁棒性，该算法应用的前提是需要使用类似 Zhang 等人[26]提出外参标定方法进行精确的多相机外参校准。

近年来，随着机器视觉技术的快速发展，视觉 SLAM 框架已相当成熟，但大部分

视觉 SLAM 算法是基于静态场景假设而设计的。当这些算法在诸如室内、街道等高动态场景中运行时，错误提取的动态物体表面的特征点将会参与位姿估计，导致定位与地图构建不准确，无疑是限制了 SLAM 算法在众多真实场景中的应用。于是，广大研究者们开始研究如何在动态环境中实现更精确的定位。

（2） 动态环境视觉 SLAM 研究现状

由于错误地提取到动态对象上的特征点会影响到 SLAM 算法的定位和建图精度，自然而然的想法是屏蔽动态区域再提取特征点或是过滤掉场景中的动态特征点，这也在学术界达成了共识。

2018 年，Bescos 等人[21]基于 ORB-SLAM2[17]框架提出了适用于动态场景的 DynaSLAM，该算法同样支持单目、双目和 RGBD 相机。该算法利用单目和双目相机进行定位时，使用 Mask-RCNN[27]对场景中可能移动的物体进行语义分割，避免 SLAM 系统提取到移动物体上的特征点；在使用 RGBD 相机时，结合多视图几何方法进行更为精准的分割。该方法以一刀切的方式舍弃了所有潜在运动物体，比如停在路边的车。因此，在存在大量潜在运动物体的场景中，该方法丢失了大量静态特征点，将导致位姿估计偏差较大。DynaSLAM 提出的同一年，Yu 等人[28]提出了 DS-SLAM。该算法在 ORB-SLAM2 基础上添加了一个独立的语义分割线程用于过滤潜在运动对象，并利用被称之为运动一致性的规则判定潜在运动区域是否真正的运动。同时，该算法还能够构建带有语义信息的地图以实现机器人更高智能程度的任务。在部分场景的实验结果表明，与 ORB-SLAM2 相比，该算法的均方根绝对轨迹精度最大提升 97.91%。2019 年，中科院研究团队提出了 Dynamic-SLAM[18]以提高动态场景下 SLAM 的定位精度。该算法结合物体运动的先验信息构建了 SSD 目标检测网络[29]用于从语义层面检测动态物体，并且设计了一个基于相邻图像速度不变性的动态物体漏检补偿算法，提高了动态物体检测召回率。同在 2019 年，Schorghuber 等人[30]利用深度学习方法将 3D 点语义整合到现有基于特征点法的 SLAM 系统中。该方法不需要运动检测，取而代之的是为每个 3D 点引入一个置信度表示物体的运动概率以判断物体是否运动，然后用语义标签的分布和 3D 点观测的一致性结合，估计 3D 点是静态点的可靠性，最后在位姿估计和优化步骤中使用这些信息。该方法可以正确区分类似停在路边的汽车和运动中的汽车这类物体。2021 年，Bescos 等人[31]改进了 DynaSLAM，提出了具备多对象跟踪能力的 DynaSLAMⅡ。该算法利用实例分割和 ORB 特征跟踪动态对象，并且使用紧耦合的方法将静态场景 3D 点、动态对象的轨迹和相机位姿进行联合优化。该算法不仅提高了 SLAM 系统在动态场景的鲁棒性还提高了机器人的场景理解能力。

上述专门针对动态场景设计的视觉 SLAM 算法，充分利用了深度学习的发展成果，将先进的目标检测算法或是语义分割算法融入到传统的视觉 SLAM 系统当中，大幅降低了 SLAM 算法在复杂动态场景中的定位漂移。但一个不可忽略的事实是，结合深度学习的 SLAM 算法的定位精度往往是以牺牲性能为代价的，大部分算法并不能在算力有限的机器人系统尤其是无人机上实时运行。总而言之，适用于动态场景的视觉

SLAM 虽取得了阶段性的研究成果，但在算力消耗、定位精度等方面还有进一步研究的空间。本文在 SLAM 算法方面的研究则旨在提升 SLAM 算法动态场景中定位精度，使之满足算力有限的无人机在复杂动态场景中进行自主飞行的定位需求。

1.2.2 运动规划研究现状

运动规划是无人机实现自主飞行关键技术之一，可大致分为路径规划（Path Planning）与轨迹规划（Trajectory Planning）。路径规划指的是在已知障碍物的条件下规划出从起始点到终点的一系列安全的离散途径点。轨迹规划指在起始点与终点之间找到一条连续的满足机器人各类约束的关于时间的函数，该函数即为机器人的连续运动轨迹。

（1） 路径规划研究现状

基于采样的路径规划算法以 RRT（Rapidly-Exploring Random Tree）算法[32]和 PRM（Probabilistic Road-Map）算法[33]为代表。RRT 算法[32]以规划起点为树的根节点，在地图里的自由空间里随机扩展构造子节点，倘若子节点扩展到终点附近，则生成一条可行路径。PRM 算法[33]从机器人的无障碍物空间进行随机采样，并构建一个无序图，然后使用图搜索算法进行路径规划。RRT 算法和 PRM 算法的一大优点是它们均具备概率完备性[34]，即在一个含有可行路径的地图中，采样次数不断增加，算法有解的概率趋近于 1。由于 RRT 算法和 PRM 算法一旦找到可行路径就不会继续迭代求解，使得它们均不具备渐进最优的特性[35]。2011 年，Karaman 等人[35]提出了 RRT*和 PRM*，改进后的算法能够在有限迭代次数后确保当前路径逼近全局最优。RRT*算法中设计了重新连接机制，将拓展树的节点进行局部重新连接，并保持根节点到每个子节点具有最短路径。同样的，PRM*在一定范围内的无序图顶点之间尝试连接。2014 年，Wu 等人[36]提出一种可变概率双向 RRT 算法（VPB-RRT），使 RRT 算法能够在无人机上应用。该算法在随机点生成过程中引入了对无人机转弯角度的限制，对空间进行栅格化，引入了覆盖率的概念，以满足对无人机路径规划的时间和精度方面的要求。2019 年，Meng 等人[37]考虑了 3 维空间的复杂性和特殊性，在改进的 RRT 算法中引入了飞行高度约束、最大航程约束和俯仰角约束，并且提出一种采样点评估标准和 3 维路径调整方法。

基于搜索的路径规划算法以 Dijkstra 算法[38]和 A*算法[39]为代表。Dijkstra 算法[38]是一种最短路径搜索算法，主要思想为寻找距离当前最近的未访问节点并将其标记为已访问，然后更新相邻节点距离，如此迭代，直到所有节点都被访问。A*算法[39]始终以启发函数$f(n)=g(n)+h(n)$最小值作为下一次访问的节点，其中$g(n)$为距离起点的代价，$h(n)$为距离终点的预计代价。Dijkstra 算法可以被看作当$h(n)=0$时 A*算法的一种特殊情况。Dijkstra 算法、A*算法和 RRT 算法、PRM 算法相比是全局最优和完备的，意味着它们在有可行路径的地图中一定有全局最优解。在后续研究中，A*等算法得到了进一步扩展和优化。2011 年，Zhuoning 等人[40]提出了一种基于虚拟力的改

进 A*算法（HVFA）。该算法给出了虚拟力混合系统模型，重新设计了路径规划方案，将其应用于无人机的路径规划时取得了良好的实时性。2016 年，Tianzhu 等人[41]提出一种改进的 3 维 A*算法，该算法将传统 A*算法从 2 维拓展到 3 维以适应复杂不确定的无人机飞行环境，改进启发函数使其满足无人机飞行约束。实验表明，该算法相比传统 A*在精确性、安全性等方面更有效。2021 年，Lim 等人[42]提出一种改进的加权 A*算法（WA*DH），该算法使用启发式角度的导数搜索生成路径最优的节点，然后搜索避开障碍物的逃逸节点，最后使用上述两步中获得的节点执行路径的局部重规划。

当前的机器人运动规划向着更高阶的方向发展，使得传统的路径规划算法往往不再被单独地应用于机器人运动规划中，但将其作为运动规划的前端搜索初始路径或是和轨迹生成方法结合构造更先进的运动规划算法不失为一种好的选择。

（2） 轨迹规划研究现状

一般来说，轨迹规划问题可以被表述为最小化目标函数问题，该目标函数由各类罚函数组合而成，如动力学约束、可行性约束和安全性约束等。2011 年，Mellinger 和 Kumar[43]提出了针对四旋翼无人机的 minimum-snap 轨迹生成算法。由于四旋翼无人机具备微分平坦特性[44]，该算法将四旋翼系统的 12 维全状态空间降低到 4 维，仅用 3D 位置和偏航角及其有限阶导数表示无人机状态，并使用样条曲线对轨迹进行参数化，最后使用轨迹对时间 4 阶导数的二范数积分作为代价函数以保证飞行轨迹的平滑度和安全性。2013 年，Richter 等人[45]并不使用二次规划（Quadratic Programming, QP）的方法求解 minimum-snap 轨迹，取而代之的是求取轨迹多项式系数的闭式解。该算法使用 RRT*算法寻找绕行障碍物的初始路径，然后基于该路径闭式求解穿越各路径点的机器人轨迹。尽管路径点是安全无碰撞的，但生成的轨迹在曲率较大处仍可能与障碍物发生碰撞，因此需要检查轨迹是否发生碰撞，在发生碰撞时通过添加中间航点重新生成局部轨迹。2015 年，Chen 等人[46]使用八叉树地图表示无人机运动规划时的安全飞行走廊，提出了一种高效的 QP 方法用于生成满足无人机高阶动态约束的无碰撞安全轨迹。2016 年，香港科技大学高飞和沈邵劼等人[47]提出一种直接在点云地图上生成无人机飞行轨迹的方法。该算法使用三维激光雷达逐步构建环境点云地图，并使用基于抽样的方法生成一系列球体组成的安全飞行走廊，然后使用二次约束二次规划生成完全约束在飞行走廊里的安全飞行轨迹。该方法虽然是概率完备的，但有时迭代周期较长，时效性较差。2018 年，Gao 和 Wu 等人[48]使用欧式符号距离场[49]（Euclidean Signed Distance Function, ESDF）表示环境地图，并使用贝塞尔（Bezier）曲线表示分段轨迹，贝塞尔曲线的凸包特性可以保证轨迹的安全性和高阶动态可行性。2020 年，Zhou 和 Gao 等人[50]提出了一种基于梯度的轨迹重规划算法并将其集成到了开源框架 Fast-Planner 中。该算法以解决传统基于梯度的轨迹优化存在局部极小值的问题。此外，Zhou 等人开发了一种拓扑路径搜索算法以探索环境中不同的可行路径，然后为每条路径进行独立的轨迹优化，此做法使得机器人对可行空间的探索更加全面。2021 年，Zhou 和 Wang 等人[51]提出一种用于多旋翼无人机的无需 ESDF 地图的轨迹规划算法，

称之为 EGO-Planner。该算法只在飞行局部区域构建栅格地图，极大程度减小了计算时间和内存消耗。该算法使用一种各向异性曲线拟合算法调整轨迹高阶导数的同时保证了轨迹形状不变，若生成的轨迹不满足机器人动态可行性，则延长该段轨迹的执行时间。大量的实验表明 EGO-Planner 在复杂环境中具有鲁棒性和实时性。随后，Zhou 等人[52,53]在 EGO-Planner 上进一步扩展和优化，提出了分布式自主导航框架 EGO-Swarm，用于在复杂障碍物环境中多机器人集群规划。2022 年，Tordesillas 等人[54]提出了一种用于动态环境的无人机实时感知轨迹规划算法，PANTHER。该算法生成的轨迹能够避开动态障碍物，同时将障碍物始终保持在视野之内，并使机器人运动造成的运动模糊降到最低以帮助跟踪动态障碍物。

从以上分析可知，运动规划算法从传统路径规划过渡到现阶段主流的轨迹规划，机器人尤其是无人机的运动规划不仅需要满足基本的无碰撞需求，还需要考虑无人机飞行的动态可行性、平滑性和实时性等。此外，诸如动态障碍物环境、集群运动规划等高阶运动规划算法正是当前研究热点和难点，还有进一步研究的必要性。本文在运动规划方面则对多旋翼无人机在复杂环境下的运动规划算法展开研究，旨在提高无人机在复杂动态环境下运动规划的安全性和鲁棒性。

1.3 论文研究内容

本文对视觉数据驱动的多旋翼无人机在复杂环境下自主飞行所涉及的其中两项关键技术——视觉 SLAM 和运动规划展开研究，具体研究内容如下：

针对视觉数据驱动的多旋翼无人机在复杂动态场景中的定位问题，提出了一种基于静态特征点筛选的动态环境 RGBD SLAM 算法。该算法利用超轻量目标检测算法在无需 GPU 的情况下实时筛选潜在动态对象。为了进一步筛选潜在动态对象区域的静态特征点，设计了一种基于高斯混合模型的静态特征点筛选算法。最后，本文基于开源 ORB-SLAM2 算法框架构建了适用于动态场景的 RGBD SLAM 算法，该算法有效提升了多旋翼无人机在复杂动态场景中的定位精度。

针对视觉数据驱动的多旋翼无人机在复杂动态场景中的运动规划问题，本文研究了基于样条曲线的多旋翼无人机局部运动规划算法。该算法使用深度相机构建局部栅格地图以减少机载计算机内存消耗，并设计了轨迹与障碍物之间的距离估计方法以指导轨迹优化。将多项罚函数的线性组合用于优化 B 样条表示的飞行轨迹。该算法提高了无人机在复杂动态场景中自主飞行的安全性与鲁棒性。

此外，基于以上研究成果，本文构建了四旋翼无人机自主飞行系统。该系统使用本文提出的动态环境 RGBD SLAM 算法作为唯一定位信息源，使用本文设计的运动规划算法生成无人机安全飞行轨迹。该系统能够在复杂动态环境中实现精准定位和安全自主飞行。

1.4 论文章节安排

本文的各章节关系如图 1-2 所示。

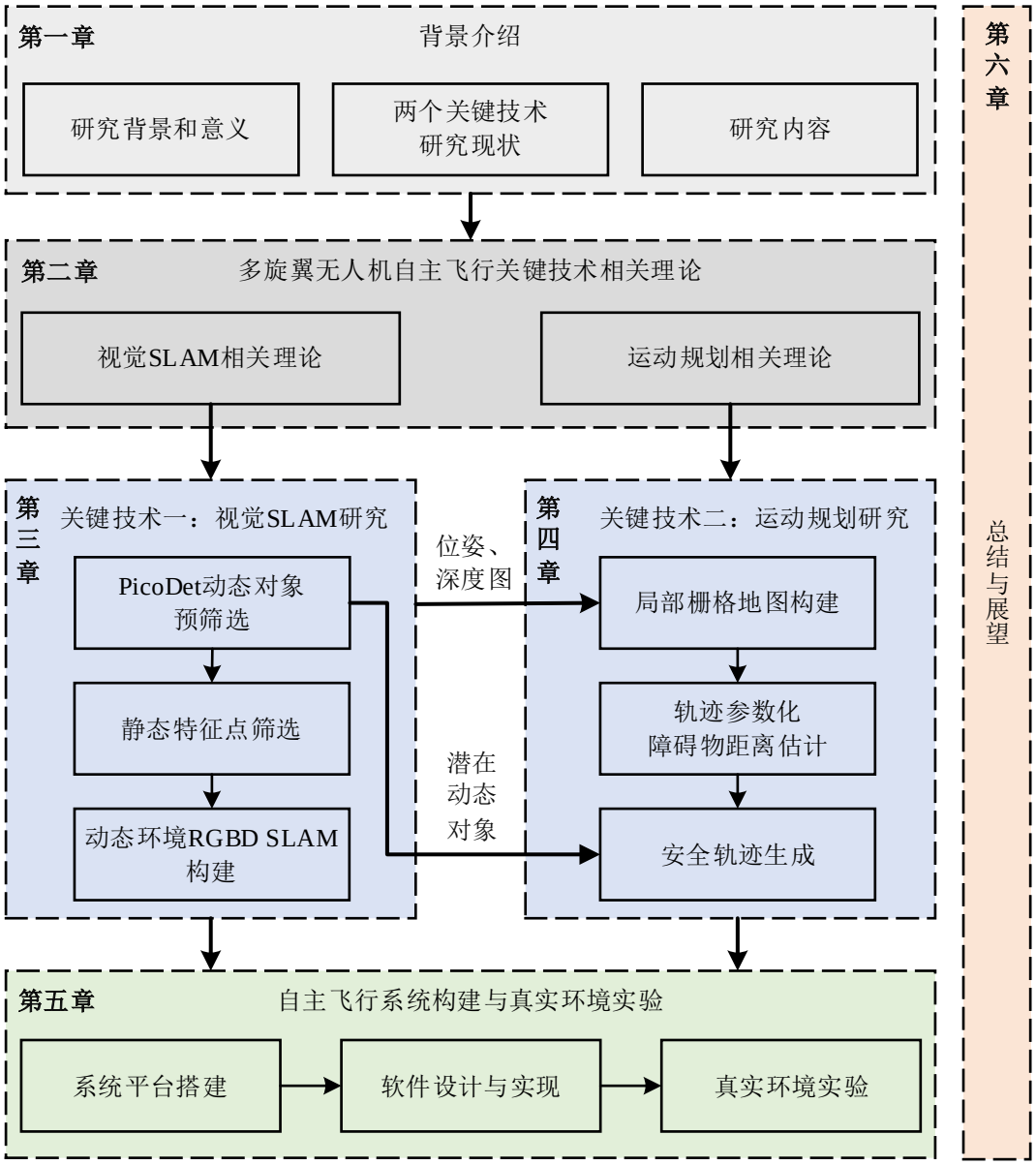


图 1-2 章节安排与论文结构

以下为各章节内容的详细描述。

第一章是论文绪论，阐述了多旋翼无人机自主飞行在多种场景中应用的重要意义和存在的挑战，然后详细介绍了多旋翼无人机自主飞行中的其中两大关键技术（SLAM 与运动规划）的国内外研究现状，并进行了分析。最后指出了本文研究内容和各章节关系。

第二章介绍了与视觉 SLAM 相关的计算机视觉方法和位姿优化理论，并给出了视觉 SLAM 的评价指标；介绍了能够简化无人机运动规划的微分平坦模型与常见的地图

表示方法。相关理论的介绍为后续章节做了铺垫。

第三章对无人机自主飞行中的关键技术之一——视觉 SLAM 技术展开研究。针对多旋翼无人机在复杂动态环境的定位问题，提出了一种基于静态特征点筛选的 RGBD SLAM 算法。首先，利用超轻量目标检测算法对潜在动态对象进行检测。然后，构建了整幅图像的光流特征高斯混合模型，利用期望最大化算法对该模型进行求解。最后，设计了一种静态特征点筛选算法进一步筛选潜在动态区域的静态特征点，以提高视觉 SLAM 的位姿估计精度，改善无人机自主飞行性能。此外，第三章算法输出的潜在动态对象检测框将参与到第四章的针对动态障碍物场景的轨迹生成中，位姿与深度图将用于运动规划中的局部栅格地图构建。

第四章对无人机自主飞行的另一关键技术——运动规划展开研究。首先，利用深度信息构建了无人机自主飞行中使用的局部栅格地图。其次，针对局部栅格地图无法表征障碍物梯度的问题，设计了算法估计飞行轨迹和障碍物之间的距离。然后，基于 B 样条对无人机飞行轨迹进行参数化，并对障碍物距离、轨迹平滑性、轨迹可行性构建罚函数，利用数值优化方法生成无人机在静态场景安全飞行轨迹。最后，利用第三章输出的潜在动态对象检测框构建简易运动模型，并利用数值优化方法生成无人机在动态场景安全飞行轨迹。

第五章对多旋翼无人机自主飞行系统进行构建和验证。对该系统硬件进行了设计，完成了机载相机的内外参标定工作；基于 ROS 机器人操作系统实现了多旋翼无人机自主飞行系统软件；利用构建的整套软硬件系统在真实环境中对本文算法进行定性和定量实验。

第六章为总结，概括本文的主要研究内容，并简要描述可能进一步开展的工作。

2 多旋翼无人机自主飞行关键技术相关理论

2.1 引言

视觉数据驱动的多旋翼无人机自主飞行涉及多个关键技术，其中包含基于视觉的 SLAM 算法、运动规划算法和无人机控制算法等，多旋翼自主飞行框架如图 2-1 所示。多旋翼无人机通过视觉传感器获取环境信息并将数据输入 SLAM 算法模块，SLAM 模块输出定位信息与环境信息，运动规划接收 SLAM 输出的定位信息以及环境信息，通过对环境建模和数值优化生成连续的安全飞行轨迹到无人机运动控制器，最后控制器以运动规划模块输出的飞行轨迹为期望值，控制无人机对该轨迹进行跟随。本论文主要关注该流程中的 SLAM 模块和运动规划模块，控制模块则采用 PixHawk[55]开源飞行控制器。

为了使后续章节中的理论分析更加具有逻辑和条理，本章将后续章节中所使用的部分重要理论进行简要介绍。首先介绍视觉 SLAM 系统中的关键核心理论，包括前端特征提取和匹配、相机投影模型、李群与李代数和后端位姿估计等。然后介绍多旋翼无人机运动规划中重要的微分平坦模型以及机器人中常用的地图表示方法。

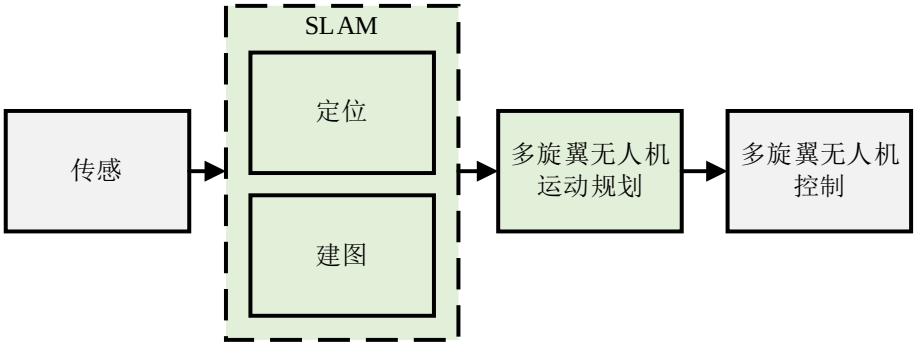


图 2-1 多旋翼无人机自主飞行框图

2.2 视觉 SLAM 相关理论

2.2.1 视觉 SLAM 问题描述

基于视觉的同时定位与建图技术主要需要完成的任务为利用视觉传感器作为输入，在没有环境先验信息的情况下，对视觉传感器观测的环境进行建模，同时估计自身的运动信息[14]。

现阶段主流的视觉 SLAM 算法主要是基于优化的方法，其算法主要流程如图 2-2 所示。视觉传感器读取模块主要完成视觉数据采集、多视觉传感器时空对齐、视觉数据灰度处理等一系列预处理操作；SLAM 前端主要工作为利用视觉传感器观测的数据通过特征提取和匹配以及根据相机投影模型估计相邻的相机运动；回环检测是根据视

觉传感器的输入判断机器人是否回到了之前的位置，若机器人回到了之前的位置，则在后端优化时可利用该信息减小位姿估计的累计误差；SLAM 后端接收前端初步估计的位姿数据和等待触发回环检测，然后利用图优化等方法对位姿进行优化，消除累计误差和提高定位精度和地图构建精度；以定位为主的 SLAM 系统中，建图模块输出的数据为路标点的点云信息。

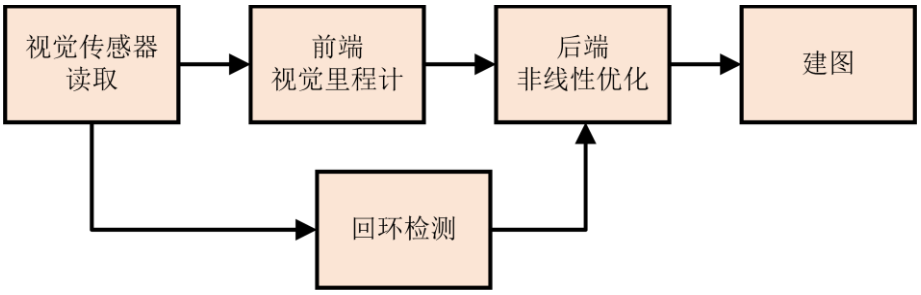


图 2-2 主流视觉 SLAM 算法流程图

2.2.2　特征提取与特征匹配

特征点检测是视觉 SLAM 算法中的第一步，良好的特征点提取算法为后续的特征匹配、位姿估计提供有力支撑。特征是一幅图像中最能代表其所表达事物特征的点和周围信息，由特征点和特征描述子组成。特征点一般是图像中灰度梯度变化明显的点，特征描述子是用数学的方法定义的特征点周围信息的数值，如此以来，在特征匹配阶段则可以通过汉明距离、马氏距离等度量方式来区分各个特征点之间的相似度，以便进行特征匹配。

ORB 特征[56]是众多特征描述方法中最有影响力的一种。ORB 特征在速度方面相较于 SIFT[57]已经有明显的提升，同时也保持了特征具有旋转与尺度不变性。ORB 特征提取算法的第一步是进行 FAST[58]（Features from Accelerated Segments Test）特征提取，FAST 特征点示意图如图 2-3 所示。FAST 特征提取算法判定中心点周围一定区域的多个像素值是否连续小于中心值，若存在连续多个像素满足条件，则认为是待选特征点。该算法通过 NMS 和特征金字塔，可以对不同尺度的图像进行特征点检测。最后 ORB 算法还通过灰度质心法计算了特征点的方向，结合后文提到的描述子计算方法保证了 ORB 特征的旋转不变性，改进之后的算法称为 Oriented-FAST。

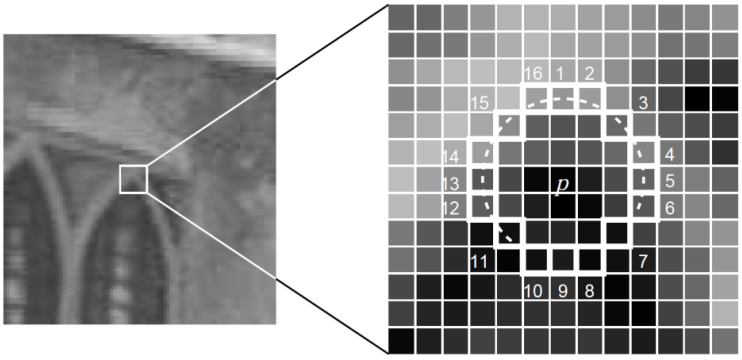


图 2-3 FAST 特征点

ORB 算法的第二步则是结合 Oriented FAST 特征点周围的信息计算特征向量，即特征描述子。ORB 使用一种名为 Rotated-BRIEF 的算法计算得到由 128 位至 512 位的“0”“1”组成的二值特征向量，提高了描述子计算效率，降低了特征匹配时间。Rotated-BRIEF 算法流程为：

（1）取特征点一定范围内的邻域；

（2）对该领域进行高斯模糊处理；

（3）对该领域进行高斯采样得到N组像素点(x,y)，然后比较每组像素值大小，$x \geqslant y$则返回1，$x < y$则返回0；

（4）步骤（3）中的N个二进制数的组合则为特征向量。

特征匹配是为了从两张图像中寻找描述同一个特征的点，也就是解决 SLAM 系统中的数据关联问题，如图 2-4 所示。为了计算两幅图像中特征之间的相似程度，可采用二进制特征向量中相同位的个数来表示。考虑在先后采样的两帧图像I_t和I_{t+1}中进行特征匹配，两帧图像中分别检测到M和N个特征$x_t^m, m \in (1, M)$和$x_{t+1}^n, n \in (1, N)$，图像I_t中的每个特征x_t^m都与图像I_{t+1}中的特征x_{t+1}^n逐一计算特征向量的相同位数，然后选择相同位数最多的特征点作为匹配点。显然，该方法的时间复杂度为$O(n^2)$，在特征点极丰富的场景中难以满足 SLAM 系统对实时性的苛刻要求。于是，现阶段快速近似最近邻（FLANN）算法[59]成为了更优的选择。

图 2-4 图像特征匹配

2.2.3 相机投影模型

（1） 针孔相机模型

相机成像是一个将$3D$空间中的点映射到$2D$图像平面上的过程。具体来说，它涉及将相机坐标系下的$3D$点通过一系列变换，最终投影到图像坐标系下的$2D$点，这个过程包括内参变换、外参变换和畸变校正等步骤。该过程相当于降维操作，这也导致了单个针孔相机的尺度不确定。针孔相机模型如图 2-5 所示。

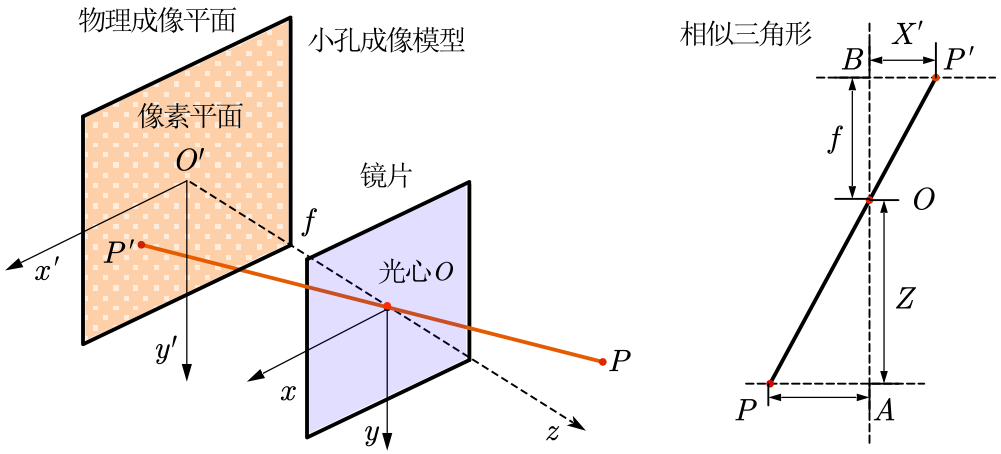


图 2-5 针孔相机模型

考虑一相机焦距为f，相机坐标系下的$3D$点$\boldsymbol{P}=[X,Y,Z]^T$，通过光心O投影到相机的物理成像平面，该点的图像坐标为$\boldsymbol{P}'=[X',Y']^T$，根据相似，如图 2-5 右图所示，可得：

$$\frac{Z}{f}=-\frac{X}{X'}=-\frac{Y}{Y'} \tag{2-1}$$

式中负号仅表示三维点在相机图像传感器上成像为倒像，而现代的相机内置的“ISP”程序会自动将图像进行旋转，因此可将式(2-1)中的符号舍去并稍加整理得：

$$\begin{cases} X' = f\dfrac{X}{Z} \\ Y' = f\dfrac{Y}{Z} \end{cases} \tag{2-2}$$

通常定义图像传感器采集的数据落在以 CMOS 芯片左上角为原点的像素坐标系里，该坐标系u、v轴分别与图像坐标系x、y轴平行。同时，与图像坐标相比像素坐标在u轴上缩放了α倍，在v轴上缩放了β倍，原点平移了$[c_x,c_y]^T$。那么，图像坐标系下坐标$\boldsymbol{P}'$与像素坐标$[u,v]^T$关系如式(2-3)：

$$\begin{cases} u=\alpha X'+c_x \\ v=\beta Y'+c_y \end{cases} \tag{2-3}$$

结合式(2-2)，可得：

$$\begin{cases} u=\alpha f\dfrac{X}{Z}+c_x \\ v=\beta f\dfrac{Y}{Z}+c_y \end{cases} \tag{2-4}$$

定义$f_x=\alpha f$，$f_y=\beta f$，并将式(2-4)整理为矩阵形式，可得：

$$Z\begin{bmatrix}u\\v\\1\end{bmatrix}=\begin{bmatrix}f_x & 0 & c_x\\0 & f_y & c_y\\0 & 0 & 1\end{bmatrix}\begin{bmatrix}X\\Y\\Z\end{bmatrix}\overset{\text{def}}{=\!=}\boldsymbol{KP} \tag{2-5}$$

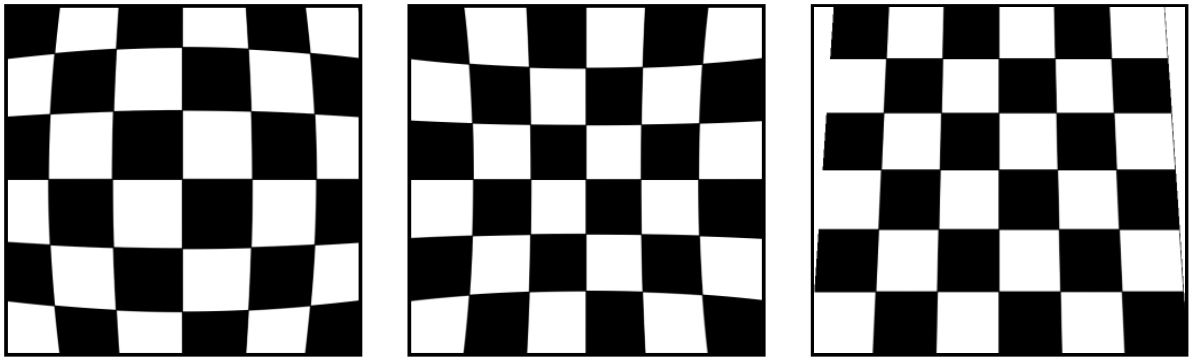
其中，$\boldsymbol{K}$ 则为相机内参矩阵。

由于相机是在三维世界中运动的，因此相机在世界坐标系下应该有三维坐标 $\boldsymbol{P}_w$，相机的旋转和平移用 $\boldsymbol{R}$ 和 $\boldsymbol{t}$ 表示，即相机的外部参数。那么世界坐标系下的一个三维点投影到像素坐标系下的关系可表示为：

$$Z\boldsymbol{P}_{uv}=Z\begin{bmatrix}u\\v\\1\end{bmatrix}=\boldsymbol{K}(\boldsymbol{R}\boldsymbol{P}_{\mathrm{w}}+\boldsymbol{t})=\boldsymbol{K}(\boldsymbol{T}\boldsymbol{P}_{\mathrm{w}})_{(1:3)} \tag{2-6}$$

（2） 畸变模型

针孔相机模型是对相机成像最简化的表达方式，实际的相机成像由于传感器与镜头安装不平行或是镜头的透镜畸变等问题难免会使最终成像产生畸变，主要有如图 2-6 所示三种，其中图 2-6(a)与图 2-6(b)属于径向畸变。



(a)桶形畸变　　(b)枕形畸变　　(c)切向畸变

图 2-6 径向畸变和切向畸变示意图

对于径向畸变，考虑归一化坐标系下的一点 $\boldsymbol{P}_c'=[x,y]^T$，其极坐标表示为 $[r,\theta]^T$，一般定义多项式对径向畸变进行描述，如式(2-7)所示：

$$\begin{cases}x_{\mathrm{dis}}=x(1+k_1r^2+k_2r^4+k_3r^6)\\y_{\mathrm{dis}}=y(1+k_1r^2+k_2r^4+k_3r^6)\end{cases} \tag{2-7}$$

其中，$[x_{dis},y_{dis}]^T$ 表示畸变之后的归一化坐标。

切向畸变需要额外的参数 p_1,p_2 进行表示：

$$\begin{cases}x_{\mathrm{dis}}=x+2p_1xy+p_2(r^2+2x^2)\\y_{\mathrm{dis}}=y+p_1(r^2+2y^2)+2p_2xy\end{cases} \tag{2-8}$$

联立式(2-7)和式(2-8)，可得通过参数 $[k_1,k_2,k_3,p_1,p_2]^T$ 表示的针孔相机畸变模型，如式(2-9)所示：

$$\begin{cases} x_{\text{dis}} = x(1 + k_1 r^2 + k_2 r^4 + k_3 r^6) + 2p_1 xy + p_2(r^2 + 2x^2) \\ y_{\text{dis}} = y(1 + k_1 r^2 + k_2 r^4 + k_3 r^6) + p_1(r^2 + 2y^2) + 2p_2 xy \end{cases} \tag{2-9}$$

至此，归纳相机投影模型步骤如图 2-7 所示。世界坐标系下一点$\boldsymbol{P}_w$通过相机外部参数$[\boldsymbol{R},\boldsymbol{t}]^T$变换到相机坐标系，并归一化得到归一化平面坐标$\boldsymbol{P}_c'$；随后将归一化平面坐标$\boldsymbol{P}_c'$经过相机畸变模型$f_1$与$f_2$，得到畸变后坐标$[x_{dis}, y_{dis}]^T$；最后通过相机内参$\boldsymbol{K}$得到像素坐标系下坐标$[u_s, v_s]^T$。

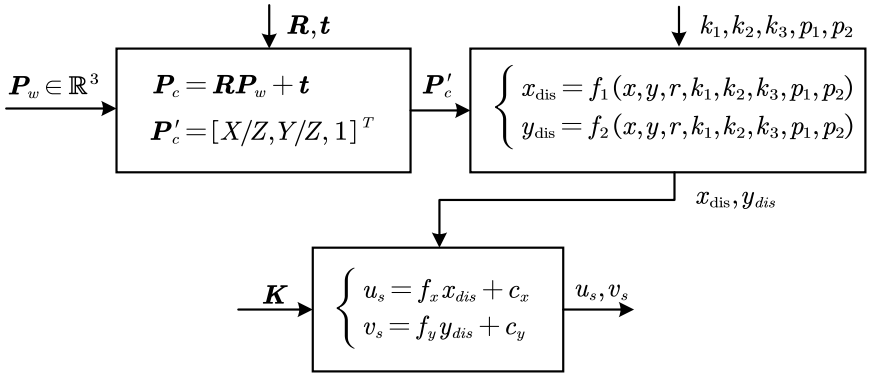


图 2-7 针孔相机投影过程

2.2.4　李群与李代数

三维世界中的刚体运动可采用旋转矩阵、欧拉角等方式进行表示，但旋转矩阵需要满足正交且行列式为 1 的约束条件，会为后续非线性优化增加困难，而欧拉角存在万向节死锁问题。为了对 SLAM 中的位姿估计进行优化，目前主流的做法是将李群转换为李代数，优化得到误差最小值后，再做李代数—李群之间的变化，以得到简洁直观的表达方式。

群是一种集合和一种运算组成的代数结构，而李群是其中较为特殊，具有连续（光滑）性质的群[60]。表示三维运动的旋转矩阵与三维变换分别构成了特殊正交群$SO(3)$和特殊欧式群$SE(3)$，表示为：

$$\begin{aligned} SO(3) &= \{\boldsymbol{R} \in \mathbb{R}^{3\times 3} | \boldsymbol{R}\boldsymbol{R}^T = \boldsymbol{I}, \det(\boldsymbol{R}) = 1\} \\ SE(3) &= \left\{ \boldsymbol{T} = \begin{bmatrix} \boldsymbol{R} & \boldsymbol{t} \\ 0^T & 1 \end{bmatrix} \in \mathbb{R}^{4\times 4} | \boldsymbol{R} \in SO(3), \boldsymbol{t} \in \mathbb{R}^3 \right\} \end{aligned} \tag{2-10}$$

李代数表示的是李群导数关系，反映了李群的局部性质。群$SO(3)$与$SE(3)$对应的李代数分别表示为：

$$\mathfrak{so}(3) = \left\{ \phi \in \mathbb{R}^3, \boldsymbol{\Phi} = \phi^\wedge = \begin{bmatrix} 0 & -\phi_3 & \phi_2 \\ \phi_3 & 0 & -\phi_1 \\ -\phi_2 & \phi_1 & 0 \end{bmatrix} \in \mathbb{R}^{3\times 3} \right\} \tag{2-11}$$

$$\mathfrak{se}(3) = \left\{ \boldsymbol{\xi} = \begin{bmatrix} \boldsymbol{\rho} \\ \phi \end{bmatrix} \in \mathbb{R}^6, \boldsymbol{\rho} \in \mathbb{R}^3, \phi \in so(3), \boldsymbol{\xi}^\wedge = \begin{bmatrix} \phi^\wedge & \boldsymbol{\rho} \\ \boldsymbol{0}^T & 0 \end{bmatrix} \in \mathbb{R}^{4\times 4} \right\} \tag{2-12}$$

其中，$\boldsymbol{\rho}$代表三维平移李代数表示，ϕ表示三维旋转李代数表示，而“$\vee$”和“$\wedge$”分别表示“反对称矩阵到向量”和“向量到反对称矩阵”的关系。李群通过对数映射可以转换为李代数，反之，李代数通过指数映射可以转换为李群。$SO(3)$与$\mathfrak{so}(3)$之间的对数映射与指数映射分别如式(2-13)、式(2-14)所示。

$$\begin{cases} \theta = \arccos\left(\dfrac{tr(\boldsymbol{R})-1}{2}\right) \\ \boldsymbol{a} = \boldsymbol{R}\boldsymbol{a} \end{cases} \tag{2-13}$$

$$\begin{aligned} \boldsymbol{R} &= \exp(\theta \boldsymbol{a}^{\wedge}) \\ &= \cos\theta \boldsymbol{I} + (1-\cos\theta)\boldsymbol{a}\boldsymbol{a}^{T} + \sin\theta \boldsymbol{a}^{\wedge} \end{aligned} \tag{2-14}$$

$SE(3)$与$\mathfrak{se}(3)$之间的对数映射与指数映射分别如式(2-15)、式(2-16)所示。

$$\begin{cases} \theta = \arccos\left(\dfrac{tr(\boldsymbol{R})-1}{2}\right) \\ \boldsymbol{a} = \boldsymbol{R}\boldsymbol{a} \\ \boldsymbol{t} = \boldsymbol{J}\boldsymbol{\rho} \end{cases} \tag{2-15}$$

$$\boldsymbol{T} = \exp(\boldsymbol{\xi}^{\wedge}) = \begin{bmatrix} \exp(\Phi^{\wedge}) & \boldsymbol{J}\boldsymbol{\rho} \\ 0^{T} & 1 \end{bmatrix} \tag{2-16}$$

其中：

$$\boldsymbol{J} = \frac{\sin\theta}{\theta}\boldsymbol{I} + \left(1-\frac{\sin\theta}{\theta}\right)\boldsymbol{a}\boldsymbol{a}^{T} + \frac{1-\cos\theta}{\theta}\boldsymbol{a}^{\wedge} \tag{2-17}$$

基于优化的 SLAM 往往需要构建一个如式(2-18)所示的优化函数，要最小化该函数，需要知道目标函数$J(\boldsymbol{T})$对优化变量$\boldsymbol{T}$的导数，以指导函数往最小值方向前进。

$$\min_{\boldsymbol{T}} J(\boldsymbol{T}) = \sum_{i=1}^{N} \| z_i - \boldsymbol{T}\boldsymbol{p}_i \|_2^2 \tag{2-18}$$

而李群只对乘法封闭，对加法不封闭，因此对李群直接求导不可行。一种解决方法是使用左扰动模型，考虑一次三维变换$\boldsymbol{R}$，对该三维变换施加微小左扰动$\Delta\boldsymbol{R}$，设左扰动$\Delta\boldsymbol{R}$对应的李代数为φ，对φ求导：

$$\begin{aligned} \frac{\partial(\boldsymbol{R}\boldsymbol{p})}{\partial\varphi} &= \lim_{\varphi\to 0}\frac{\exp(\varphi^{\wedge})\exp(\phi^{\wedge})\boldsymbol{p} - \exp(\phi^{\wedge})\boldsymbol{p}}{\varphi} \\ &= \lim_{\varphi\to 0}\frac{(\boldsymbol{I}+\varphi^{\wedge})\exp(\phi^{\wedge})\boldsymbol{p} - \exp(\phi^{\wedge})\boldsymbol{p}}{\varphi} \\ &= \lim_{\varphi\to 0}\frac{\varphi^{\wedge}\boldsymbol{R}\boldsymbol{p}}{\varphi} = \lim_{\varphi\to 0}\frac{-(\boldsymbol{R}\boldsymbol{p})^{\wedge}\varphi}{\varphi} = -(\boldsymbol{R}\boldsymbol{p})^{\wedge} \end{aligned} \tag{2-19}$$

这便是$SO(3)$上对扰动量的导数。同理，可推出$SE(3)$上对扰动量$\Delta\boldsymbol{T}$的李代数$\delta\boldsymbol{\xi}$的导数为：

$$\begin{aligned}\frac{\partial(\boldsymbol{Tp})}{\partial\delta\boldsymbol{\xi}}&=\lim_{\delta\boldsymbol{\xi}\to0}\frac{\exp(\delta\boldsymbol{\xi}^{\wedge})\exp(\boldsymbol{\xi}^{\wedge})\boldsymbol{p}-\exp(\boldsymbol{\xi}^{\wedge})\boldsymbol{p}}{\delta\boldsymbol{\xi}}\\&=\lim_{\delta\boldsymbol{\xi}\to0}\frac{\begin{bmatrix}\delta\phi^{\wedge}(\boldsymbol{Rp}+\boldsymbol{t})+\delta\boldsymbol{\rho}\\0^{T}\end{bmatrix}}{[\delta\boldsymbol{\rho},\delta\boldsymbol{\phi}]^{T}}=\begin{bmatrix}\boldsymbol{I}&-(\boldsymbol{Rp}+\boldsymbol{t})^{\wedge}\\0^{T}&0^{T}\end{bmatrix}\end{aligned}\tag{2-20}$$

2.2.5　位姿估计

本论文所使用的 RGBD 相机由于具有深度信息，可以获得特征点的3D位置，因此不需要依赖对极几何[61]，可直接通过 PnP[62]（Perspective-n-Point）进行求解位姿。PnP 求解方法中最重要之一当属光束法平差[63]（Bundle Adjustment, BA）。BA 将相机在世界坐标系的位姿和路标点同时进行优化，构建路标点观测值与路标点重投影值（即根据特征匹配将t时刻的路标点通过相机投影模型投影到$t+1$时刻的图像中）之差的误差项，然后对误差进行累加建立最小二乘问题，如式(2-21)所示：

$$\boldsymbol{T}^{*}=\arg\min_{\boldsymbol{T}}\frac{1}{2}\left\|\boldsymbol{u}_i-\frac{1}{s_i}\boldsymbol{K}\underbrace{(\boldsymbol{TP}_{wi})_{(1:3)}}_{\boldsymbol{P}_i}\right\|_2^2\tag{2-21}$$

其中，s_i表示第i个特征点的深度，$\boldsymbol{u}_i$表示第i个特征点像素坐标，$\boldsymbol{K}$为相机内参，$\boldsymbol{T}$为相机外参的齐次表示，$\boldsymbol{P}_{wi}=[X_w,Y_w,Z_w,1]^T$为3$D$点齐次坐标，为了方便表示，$(\boldsymbol{TP}_{wi})_{(1:3)}$表示取矩阵$\boldsymbol{TP}_w$的前三行三列，变为非齐次坐标形式，即$\boldsymbol{P}_i$。

为了对该最小二乘问题进行优化，还需要求得误差项对优化变量导数的解析形式。根据相机模型式，可得：

$$u=f_x\frac{X}{Z}+c_x,\ v=f_y\frac{Y}{Z}+c_y\tag{2-22}$$

利用李代数左扰动模型对位姿$\boldsymbol{T}$施加微小扰动$\delta\boldsymbol{\xi}$，求误差相当于扰动量的变化，利用归一化坐标$\boldsymbol{P}$做中间变量，根据链式求导法则可得：

$$\frac{\partial\boldsymbol{e}}{\partial\delta\boldsymbol{\xi}}=\lim_{\delta\boldsymbol{\xi}\to0}\frac{\boldsymbol{e}(\delta\boldsymbol{\xi}\oplus\boldsymbol{\xi})-\boldsymbol{e}(\boldsymbol{\xi})}{\delta\boldsymbol{\xi}}=\frac{\partial\boldsymbol{e}}{\partial\boldsymbol{P}}\frac{\partial\boldsymbol{P}}{\partial\delta\boldsymbol{\xi}}\tag{2-23}$$

其中，符号$\oplus$代表左扰动。误差相对于$\boldsymbol{P}$的导数根据式(2-22)可得：

$$\frac{\partial\boldsymbol{e}}{\partial\boldsymbol{P}}=-\begin{bmatrix}\frac{\partial u}{\partial X}&\frac{\partial u}{\partial Y}&\frac{\partial u}{\partial Z}\\\frac{\partial v}{\partial X}&\frac{\partial v}{\partial Y}&\frac{\partial v}{\partial Z}\end{bmatrix}=-\begin{bmatrix}\frac{f_x}{Z}&0&-\frac{f_xX}{Z^2}\\0&\frac{f_y}{Z}&-\frac{f_yY}{Z^2}\end{bmatrix}\tag{2-24}$$

而误差相对于李代数的导数在式(2-20)中已给出，取其前三维与式(2-24)相乘可得误差相对于优化变量的 Jacobian 矩阵，如式(2-25)所示。该 Jacobian 矩阵便可指导非线性优化朝着极小值方向前进。

$$\frac{\partial \boldsymbol{e}}{\partial \delta \boldsymbol{\xi}} = -\begin{bmatrix} \frac{f_x}{Z} & 0 & -\frac{f_x X}{Z'^2} & -\frac{f_x XY}{Z^2} & f_x + \frac{f_x X^2}{Z^2} & -\frac{f_x Y}{Z} \\ 0 & \frac{f_y}{Z} & -\frac{f_y Y}{Z^2} & -f_y - \frac{f_y Y^2}{Z^2} & \frac{f_y XY}{Z^2} & \frac{f_y X}{Z} \end{bmatrix} \tag{2-25}$$

2.2.6 视觉 SLAM 评价指标

SLAM 算法的精度评价指标主要有绝对轨迹误差（Absolute Trajectory Error, ATE）和相对位姿误差（Relative Pose Error, RPE）两种[64]。ATE 评估的是算法输出的估计位姿与真实值之间的误差，可以反映算法定位精度和轨迹的全局一致性，而 RPE 描述固定时间间隔为Δt的两帧估计值与真实值之间的位姿精度。式(2-26)为利用均方根误差（RMSE）计算 ATE 和 PRE 方法，除此之外，还可以利用平均数、中位数、标准差等方式来对误差进行评估。

$$\begin{aligned} \mathrm{ATE} &= \sqrt{\frac{1}{N}\sum_{i=1}^{N} \| \log(\boldsymbol{T}_{\mathrm{gt},i}^{-1}\boldsymbol{T}_{\mathrm{est},i})^{\vee} \|_2^2} \\ \mathrm{RPE} &= \sqrt{\frac{1}{N-\Delta t}\sum_{i=1}^{N-\Delta t} \| \log((\boldsymbol{T}_{\mathrm{gt},i}^{-1}\boldsymbol{T}_{\mathrm{gt},i+\Delta t}))^{-1}(\boldsymbol{T}_{\mathrm{est},i}^{-1}\boldsymbol{T}_{\mathrm{est},i+\Delta t}))^{\vee} \|_2^2} \end{aligned} \tag{2-26}$$

其中，N 表示位姿数量，$\boldsymbol{T}_{gt}$ 表示真实值，$\boldsymbol{T}_{est}$ 表示估计值。

2.3 多旋翼无人机运动规划相关理论

2.3.1 运动规划问题描述

运动规划是指在机器人构型空间[65]下生成一条满足约束的安全轨迹，对于多旋翼无人机的运动规划来说，该约束可以为与障碍物的无碰撞约束、无人机动力学的约束、轨迹可行性约束、能量最小约束等。

大部分运动规划算法分为前端路径搜索和后端轨迹生成，运动规划算法使用各类路径搜索算法得到初始路径，再将离散路径参数化为多项式初始轨迹，最后结合各类约束对初始轨迹进行优化，得到最终满足约束的安全飞行轨迹。其总体框图如图 2-8 所示。

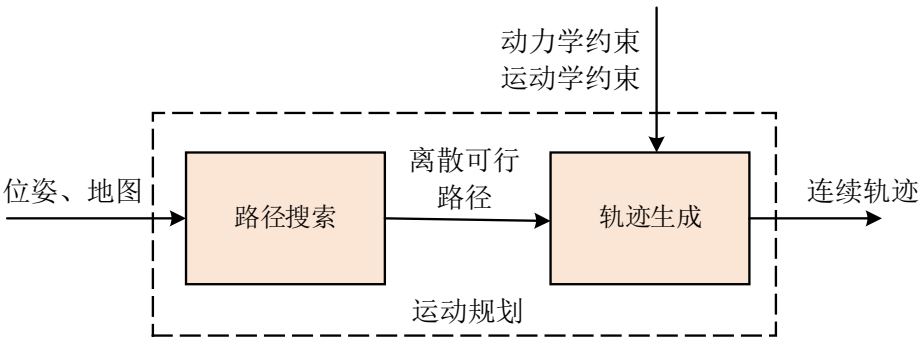


图 2-8 运动规划框图

2.3.2　微分平坦模型

1995 年，Fliess 等人[44]首次提出了微分平坦的概念：如果一个动力系统的全状态和控制输入可以被其平坦输出及其有限阶导数唯一确定，则该系统为微分平坦系统。2011 年，Mellinger 和 Kumar 等人[43]验证了平行轴四旋翼无人机为微分平坦系统，并给出了四旋翼无人机平坦输出，如式(2-27)所示：

$$\sigma = [x, y, z, \psi]^T \tag{2-27}$$

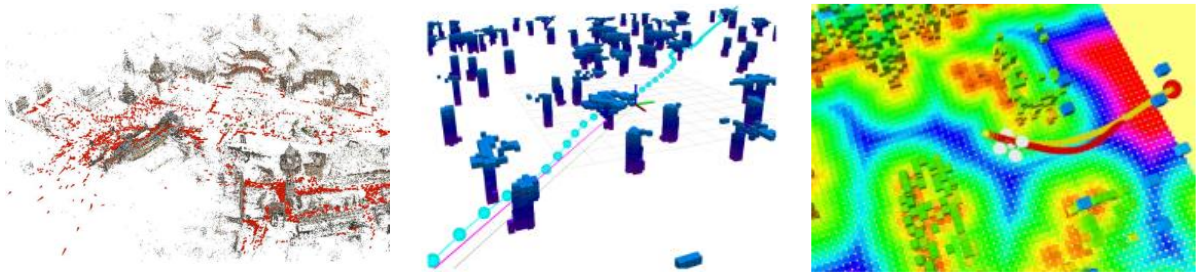
式中，$[x, y, z]^T$ 为无人机质心的$3D$坐标，ψ 为无人机的偏航角。那么定义在平坦输出空间的一条光滑轨迹$\sigma(t)$可以表示为：

$$\sigma(t):[t_0, t_m] \to \mathbb{R}^3 \times SO(2) \tag{2-28}$$

也就是说，四旋翼的全状态$\mathbf{X} = [x, y, z, \phi, \theta, \psi, \dot{x}, \dot{y}, \dot{z}, \omega_x, \omega_y, \omega_z]^T$可以由平坦输出$\sigma$及其有限阶微分表示，无人机在进行轨迹规划时，只用对状态变量σ进行规划，即只需规划无人机的位置与偏航角，从而避免规划四旋翼全状态带来的困难。Mellinger 和 Kumar 等人[43]给出了使用微分平坦输出及其有限阶导数表示四旋翼全状态的方法。也正是因为四旋翼无人机具有微分平坦特性，使得轨迹规划中最优化的维度从 12 维降低为 4 维，极大提高了算法的实时性，同时，该特性也保证了轨迹$\sigma(t)$满足四旋翼无人机的动力学约束。

2.3.3　移动机器人中地图的表示

移动机器人中常用的地图如图 2-9 所示。



(a) 点云地图　　(b) 栅格地图　　(c) ESDF地图

图 2-9 移动机器人常用地图类型示例

点云地图由一系列无序离散点的集合，可由 RGBD 相机高效生成，保留了完整的数据信息，但存储量大、配准计算量大，同时由于点云地图各点之间无任何数据关联，障碍物某个位置的可通过性未知，通常不直接用于机器人导航。

栅格地图以一定的分辨率将连续的物理空间转变为离散的栅格，每个栅格有占据和非占据两种状态，运动规划系统可以直接判断某个栅格的占据状态以确定该区域是否为障碍物。在实际应用中可以根据需要对分辨率进行调整以调整数据量大小，其数据规模小于点云地图，同时栅格地图利用概率表示每个栅格的占据状态，每次传感器数据到达时将会更新栅格占据概率，解决了点云地图不能动态更新的问题。

欧式符号距离场（Euclidean Signed Distance Function, ESDF）中每个元素存储了距

离最近障碍物的距离，因此可以指导运动规划生成的轨迹远离障碍物。开源的Voxblox[66]、FIESTA[67]和 Voxfield[68]都能够快速增量构建 ESDF 地图。

2.4 本章小结

本章对视觉 SLAM 和无人机运动规划中的相关理论进行了梳理。关于视觉 SLAM，首先介绍了视觉 SLAM 中 ORB 特征提取与匹配方法，简略的推导了简单针孔相机的投影模型并给出了相机的畸变模型，然后描述了李群和李代数之间的相互映射关系，再结合本章前置知识，构造了位姿估计的最小二乘问题，并给出了误差对优化变量导数的解析形式，最后介绍了视觉 SLAM 关于精度的评价指标；关于运动规划，介绍了平行轴四旋翼无人机的微分平坦特性和移动机器人中常用的地图表示方式。

3　基于静态特征点筛选的动态环境 RGBD SLAM

3.1　引言

SLAM 是无人机自主飞行关键技术之一。传统视觉 SLAM 近年来得到了长足发展，在大部分结构化场景中能够取得良好的定位精度，但对于复杂、尤其是包含动态对象的场景仍然存在诸多挑战。在动态场景中，针对静态场景设计的视觉 SLAM 算法会错误地提取到与动态对象相关联的特征点，并将其用于相机的位姿估计，这无疑会增加 SLAM 算法在动态场景中的定位误差。虽然传统视觉 SLAM 未对场景中动态特征进行特殊处理，但位姿估计中通常会使用随机采样一致性算法[69]排除外点影响，因此在包含少量低速移动物体的场景中仍然具有鲁棒性，但对高动态场景依然束手无策。对于无人机自主飞行中存在动态障碍物的场景，倘若无人机的位姿估计存在较大偏差，无人机将会跟随错误的飞行轨迹，导致无人机飞行安全性将得不到保障。

本章针对无人机自主飞行中存在动态障碍物的场景，提出了一种适用于无人机自主飞行的动态环境 SLAM 算法，结构如图 3-1 所示。本章第 3.2 节利用神经网络检测潜在动态对象。为了进一步筛选检测框中的静态特征点，使更多静态特征点参与后端位姿估计，本章第 3.3 节利用光流法对特征点进行跟踪，计算特征点光流的方向和二范数，然后将光流特征建模为高斯混合模型[70]（Gaussian Mixed Model, GMM），并利用期望最大化（Expectation Maximization, EM）算法[71]对该模型进行聚类。结合 GMM 的聚类结果，设计了静态特征点筛选算法，并将该算法集成到先进的 ORB-SLAM2[17] 中进行实验。结果表明本章算法能够有效筛选出动态场景中的静态特征点，改进后的 ORB-SLAM2 算法对动态场景鲁棒且计算复杂度低，不依赖 GPU，可在机载计算平台实时运行。

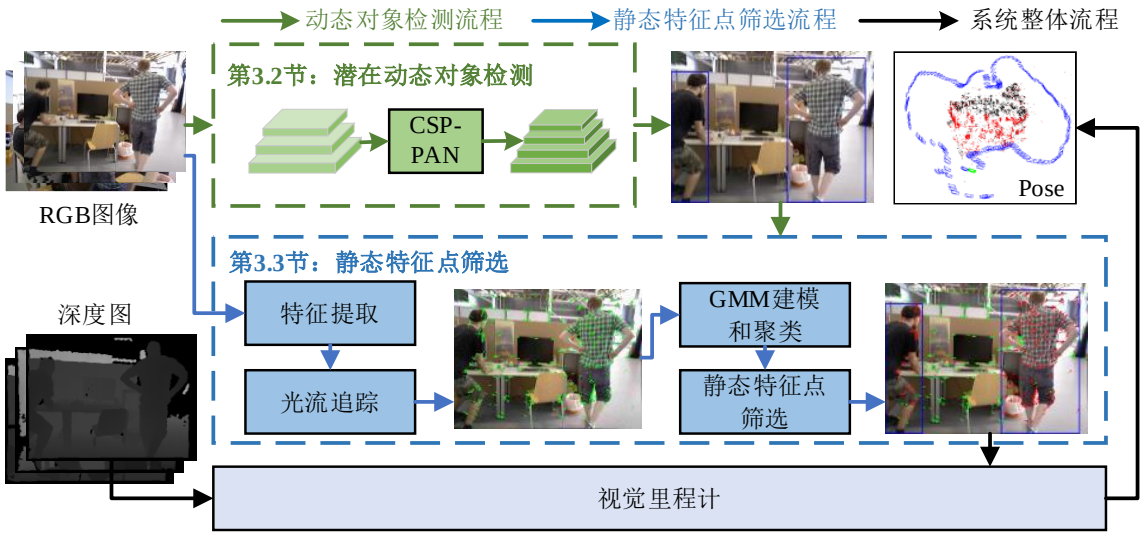


图 3-1 算法整体结构图

3.2 引入超轻量网络的潜在动态对象检测

目标检测是计算机视觉的经典任务之一，旨在预测图像中目标的类别和该目标的像素坐标。为了满足视觉 SLAM 在 PC 和机载计算机运行的实时性，在目标检测网络的选择上需要对预测精度和推理速度进行权衡。显然，在 SLAM 应用中，推理速度比预测精度具有更高的优先级。

考虑到 SLAM 算法需要在无人机载计算机上实时运行，选择超轻量目标检测算法 PicoDet[72]对视觉范围内的潜在运动对象进行检测，PicoDet 网络结构如图 3-2 所示。

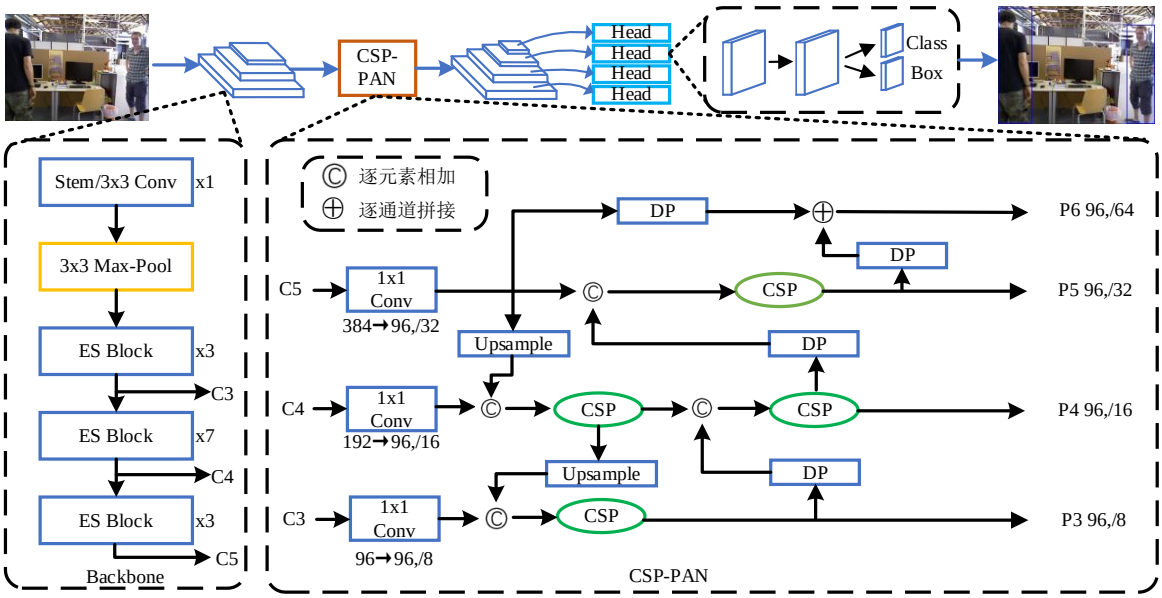


图 3-2 PicoDet 网络结构

PicoDet 同大部分目标检测网络一样，包含用于提取特征的 Backbone 网络、用于特征融合的 Neck 网络、用于信息预测的 Head 网络三个部分组成。PicoDet 的骨干网络为对 ShuffleNetv2[73]的改进版本 Enhanced ShuffleNet(ES-Net)，能够在减少模型大小的同时尽可能多的提取图像特征，颈部引入 CSP 结构[74,75]和 PAN 结构[76]进行特征聚合，检测头则使用 5×5 空洞卷积扩大感受野，提高预测精度。该算法激活函数使用 HSwish[77]代替常用的 ReLU，其激活函数如式(3-1)所示。

$$\mathrm{HSwish}(x)=x\frac{\mathrm{ReLU}6(x+3)}{6} \tag{3-1}$$

其中，$\mathrm{ReLU}6(x)=\min(6,\max(0,x))$ 为最大值为 6 的 ReLU 激活函数。该激活函数存在上界，但不存在下界，可以避免梯度饱和以及提供更强的正则化效果。

3.2.1 骨干网络 ES-Net

为了使骨干网络更关注通道间的重要特征，在网络中引入 SE(Squeeze and Excitation)模块[78]，通过显式建模，让网络自适应的对通道重要程度进行校准，SE 模块结构如图 3-3 所示。

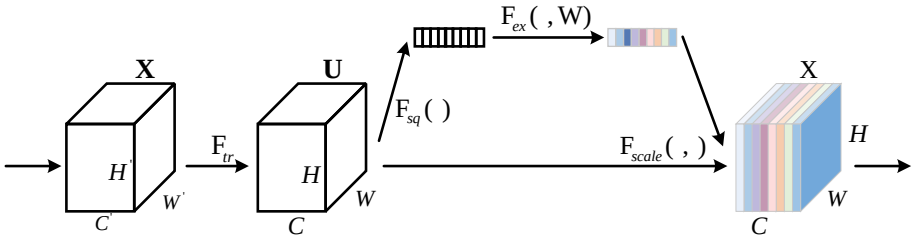


图 3-3 SE 模块结构

首先，SE 模块对网络进行压缩（Squeeze），将全局空间信息压缩到一个通道描述符中，通过全局平均池化统计通道信息，将二维的特征信息变为一维实数，如式(3-2)所示，该实数能够反映通道间的全局信息。

$$z_c = F_{sq}(u_c) = \frac{1}{H \times W} \sum_{i=1}^{H} \sum_{j=1}^{W} u_c u_c(i,j) \tag{3-2}$$

然后，SE 模块对网络进行激活（Excitation）校准，为了充分利用前一步操作压缩得到的全局信息z_c，设计了基于σ激活函数的门控机制：

$$s_c = F_{ex}(z_c, \mathbf{W}) = \sigma(g(z_c, \boldsymbol{W})) = \sigma(\mathbf{W}_2 \delta(\mathbf{W}_1 z_c)) \tag{3-3}$$

其中$\boldsymbol{W}_1 \in \mathbb{R}^{\frac{C}{r} \times C}$，$\boldsymbol{W}_2 \in \mathbb{R}^{C \times \frac{C}{r}}$，$r$为压缩比率，$\delta$为 ReLU 激活函数。经过该门控机制后，输出参数s_c表示通道c的权重，校准后通道c的特征可以由权重s_c与对应的特征通道相乘得到，如式(3-4)所示。

$$\tilde{\boldsymbol{x}}_c = F_{\text{scale}}(\mathbf{u}_c, s_c) = s_c \cdot \mathbf{u}_c \tag{3-4}$$

骨干网络 ES-Net 利用 ShuffleNetv2 的通道洗牌（Channel Shuffle），获得了通道间信息交换的功能，却丢失了通道间特征融合功能。为了解决这一问题，在步进为 2 的网络中引入了深度可分离卷积以融合不同通道的信息，在步进为 1 的网络中引入 Ghost 模块[79]以较小的参数量获得更多的特征图。ES-Net 骨干网络结构如图 3-4 所示。

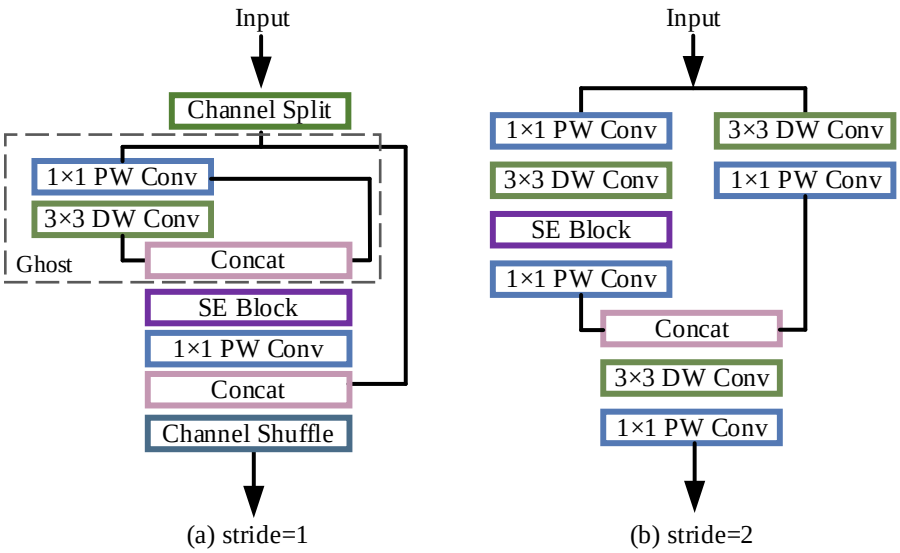


图 3-4 ES-Net 网络结构

3.2.2 PicoDet 颈部与检测头

PicoDet 的 Neck 网络包含 4 个分支，使用 PAN[76]获取多层次的特征图，并利用在 YOLOv4[74]和 YOLOX[75]中广泛使用的 CSP 结构进行特征融合。在原始的 CSP-PAN 结构中，输入通道的数量与骨干网络的输出保持一致，大量的通道数会导致较大的计算成本，不利于边缘平台推理。在 PicoDet 颈部输入时通过1×1卷积将输入通道数降低为 96，然后通过 CSP 结构进行自上而下和自下而上的特征融合，既降低了模型计算量，又保证了推理精度。除此之外，除了输入时改变通道数的1×1卷积，颈部其余卷积均采用了深度可分离卷积以尽可能多的降低模型参数量。

Head 网络同样含有 4 个分支，并分别与颈部的 4 个分支相连接。虽然没有像 YOLOX 一样采用减少通道数量的解耦检测头换取更高的精度，但是 PicoDet 在各通道采用较多的深度可分离卷积，使得 PicoDet 在减少参数量大小的同时仍然保留了较好的检测精度。

3.2.3 模型部署

本文选用的 PicoDet 在微软 COCO 数据集上进行测试，$mAP(0.5:0.95)$为 27.1，$mAP(0.5)$达 41.4，与轻量级模型 YOLOv4-Tiny、YOLOX-Tiny 相比，均处于领先水平。考虑到 PicoDet 需要集成到 SLAM 算法中以及机载计算机无高性能 GPU，本文首先对 PicoDet 全精度模型进行 FP16 量化，网络模型大小仅为$2.26M$。然后利用腾讯 NCNN[80]神经网络前向计算框架将模型部署在 CPU，在 Ryzen7 R4800H 推理，帧率为 55FPS，在机载计算机（CPU 为 I7-8565U）进行推理，帧率达 40FPS，满足 SLAM 实时性需要。

在本文中，潜在运动对象为预先设定的多个类别，包含人、自行车、车辆等，使用 PicoDet 在 TUM RGBD 数据集上进行潜在运动对象筛选示例如图 3-5 所示。当然，若自主飞行场景发生变化，也可以删减或者增加类别以实现潜在运动对象筛选且并不影响筛选性能。

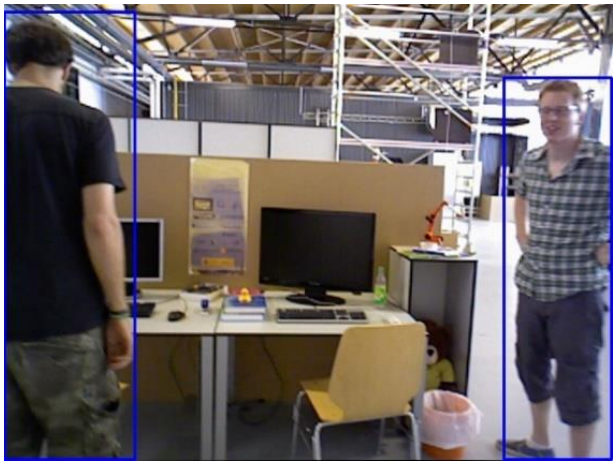


图 3-5 PicoDet 进行潜在动态物体筛选

3.3　潜在动态对象中的静态特征点筛选

在 3.2 节中，本论文通过超轻量目标检测网络 PicoDet 对潜在动态物体进行了预筛选。为了进一步筛选潜在动态对象区域的静态特征点，使更多特征点参与位姿估计，本节着重介绍基于光流特征高斯混合模型的静态特征点筛选算法。

3.3.1　ORB 特征点光流跟踪

光流法旨在第二幅图像中寻找第一幅图像中特征点的位置，常用于运动估计、SLAM 前端特征点跟踪。对图像中的每个像素都进行跟踪通常称为稠密光流，仅跟踪图像中部分像素点，被称为稀疏光流。本文使用稀疏光流对 ORB 特征点进行跟踪，在特征点运动求解子问题则采用 LK 法，称为 LK 光流[81]。

LK 光流使用$\boldsymbol{I}(t)$表示图像关于时间的函数，认为图像是随时间变化的，如图 3-6 所示。

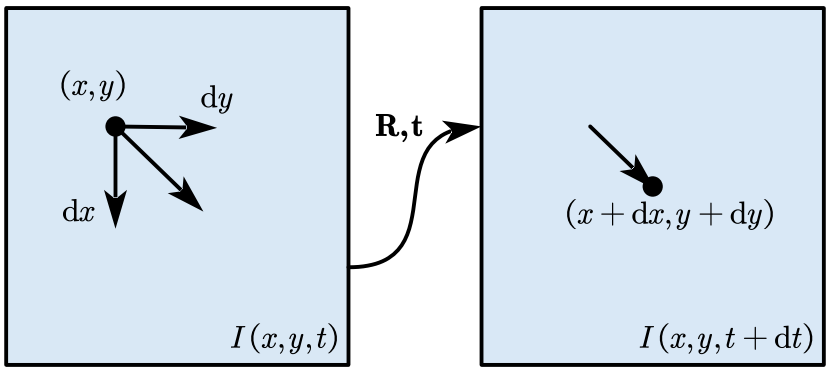


图 3-6 LK 光流示意图

在t时刻图像$\boldsymbol{I}$中(x,y)处的灰度值表示为$\boldsymbol{I}(x,y,t)$。LK 光流基于以下三点假设：

（1）灰度不变性：一个具体的三维空间点在每次图像采样时的灰度值是恒定的；

（2）空间一致性：三维空间中相同表面相邻点的运动趋势相近，变换到图像坐标系下距离也较近；

（3）时间持续性：时间的变化不会导致特征点位置的剧烈变化。

由以上三点假设可知，LK 光流在光线变化缓慢、运动较慢或相机帧率较高的场景中更为适用。

对于t时刻图像$\boldsymbol{I}$中(x,y)处像素，设其在$t+\mathrm{d}t$时刻运动到$(x+\mathrm{d}x,y+\mathrm{d}y)$处，基于灰度不变性与时间持续性假设，有：

$$\boldsymbol{I}(x+\mathrm{d}x,y+\mathrm{d}y,t+\mathrm{d}t)=\boldsymbol{I}(x,y,t) \tag{3-5}$$

对式(3-5)左边进行泰勒展开，得：

$$\begin{aligned}\boldsymbol{I}(x+\mathrm{d}x,y+\mathrm{d}y,t+\mathrm{d}t)=\boldsymbol{I}(x,y,t)+\frac{\partial \boldsymbol{I}}{\partial x}\mathrm{d}x+\frac{\partial \boldsymbol{I}}{\partial y}\mathrm{d}y+\frac{\partial \boldsymbol{I}}{\partial t}\mathrm{d}t\\+R(x,y,t)\end{aligned} \tag{3-6}$$

其中，$R(x,y,t)$为佩亚诺（Peano）余项，近似为0。

联立式(3-5)与式(3-6)，可得：

$$\frac{\partial \boldsymbol{I}}{\partial x}\mathrm{d}x+\frac{\partial \boldsymbol{I}}{\partial y}\mathrm{d}y+\frac{\partial \boldsymbol{I}}{\partial t}\mathrm{d}t=0 \tag{3-7}$$

式(3-7)两边同时除以$\mathrm{d}t$，有：

$$\frac{\partial \boldsymbol{I}}{\partial x}\frac{\mathrm{d}x}{\mathrm{d}t}+\frac{\partial \boldsymbol{I}}{\partial y}\frac{\mathrm{d}y}{\mathrm{d}t}+\frac{\partial \boldsymbol{I}}{\partial t}=0 \tag{3-8}$$

其中，$\frac{\mathrm{d}x}{\mathrm{d}t}$和$\frac{\mathrm{d}y}{\mathrm{d}t}$分别表示$(x,y)$处像素在$x$和$y$方向的运动速度，分别记为$u$和$v$。$\frac{\partial \boldsymbol{I}}{\partial x}$与$\frac{\partial \boldsymbol{I}}{\partial y}$为$(x,y)$处像素的梯度，记为$\boldsymbol{I}_x$、$\boldsymbol{I}_y$，实际使用中，ORB特征点周围梯度为已知，一般选取当前帧的像素梯度。$\frac{\partial \boldsymbol{I}}{\partial t}$为灰度值关于时间的导数，记为$\boldsymbol{I}_t$。将式(3-8)写为矩阵形式：

$$\begin{bmatrix}\boldsymbol{I}_x & \boldsymbol{I}_y\end{bmatrix}\begin{bmatrix}u\\v\end{bmatrix}=-\boldsymbol{I}_t \tag{3-9}$$

式(3-9)为二元一次方程，一个特征点基于灰度不变性的约束不足以计算该方程。基于空间不变性假设，考虑$m\times m$的相邻区域，共m^2个约束方程：

$$\begin{bmatrix}\boldsymbol{I}_{x1} & \boldsymbol{I}_{y1}\\ \boldsymbol{I}_{x2} & \boldsymbol{I}_{y2}\\ \vdots & \vdots\\ \boldsymbol{I}_{xn} & \boldsymbol{I}_{yn}\end{bmatrix}\begin{bmatrix}u\\v\end{bmatrix}=\begin{bmatrix}-\boldsymbol{I}_{t1}\\ -\boldsymbol{I}_{t2}\\ \vdots\\ -\boldsymbol{I}_{tm}\end{bmatrix} \tag{3-10}$$

另$\boldsymbol{A}=\begin{bmatrix}\boldsymbol{I}_{x1} & \boldsymbol{I}_{y1}\\ \boldsymbol{I}_{x2} & \boldsymbol{I}_{y2}\\ \vdots & \vdots\\ \boldsymbol{I}_{xn} & \boldsymbol{I}_{yn}\end{bmatrix}$，$\boldsymbol{b}=\begin{bmatrix}-\boldsymbol{I}_{t1}\\ -\boldsymbol{I}_{t2}\\ \vdots\\ -\boldsymbol{I}_{tm}\end{bmatrix}$，则式(3-10)可简化为：

$$\boldsymbol{A}\begin{bmatrix}u\\v\end{bmatrix}=\boldsymbol{b} \tag{3-11}$$

该式为超定方程，可采用最小二乘法求解：

$$\begin{bmatrix}\hat{u}\\v\end{bmatrix}=-(\boldsymbol{A}^{\mathrm{T}}\boldsymbol{A})^{-1}\boldsymbol{A}^{\mathrm{T}}\boldsymbol{b} \tag{3-12}$$

至此，计算得到的$[u,v]^T$为特征点在x和y方向的移动速度，由于相机帧率已知，则可以计算得到当前帧的 ORB 特征点在下一帧图像中的位置。图 3-7 给出了利用 LK 光流算法得到的 ORB 特征点跟踪效果，其中绿色线条即为 ORB 特征点光流。

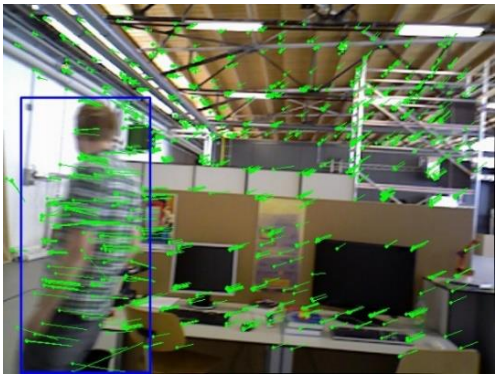
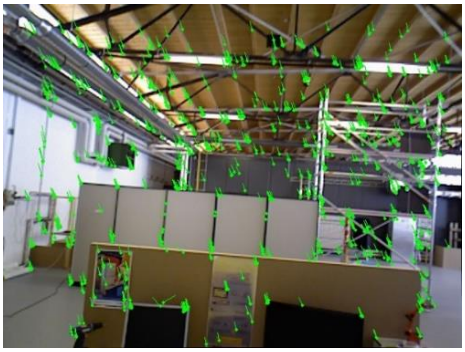


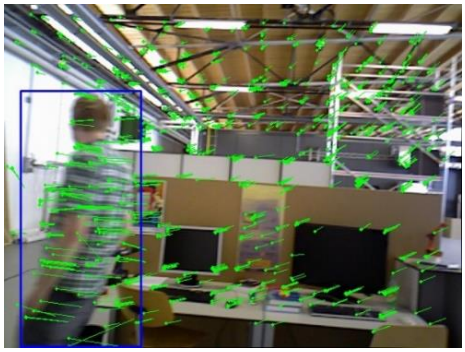
图 3-7 LK 光流特征点跟踪效果

3.3.2　光流特征高斯混合模型构建与求解

在实验中，我们发现使用 LK 光流对静态场景的特征点进行追踪时，光流的方向和二范数在两个维度或者一个维度具有一定相似性，具体表现为静态对象表面的光流方向与光流二范数数值接近，而动态对象表面的光流方向与光流二范数与静态对象表面数值差异较大，如图 3-8 所示。



(a) 静态场景光流



(b) 动态场景光流

图 3-8 静态场景与动态场景光流特点对比

直接的，定义光流的方向与二范数作为后续高斯混合模型的二维特征向量，定义光流方向为：

$$l=\left\|\overrightarrow{p_1p_2}\right\|_2=\sqrt{(x_1-x_2)^2+(y_1-y_2)^2} \tag{3-13}$$

其中 $p_1=(x_1,y_1)$ 为当前图像帧特征点坐标，$p_2=(x_2,y_2)$ 为相邻帧特征点坐标。

为了尽量将特征降维，本文不考虑直接使用特征点光流 2 维方向向量表示方向，而是将光流方向变换为角度进行表示，定义如下：

$$\theta=\begin{cases}\operatorname{acos}\left(\dfrac{x_2-x_1}{\left\|\overrightarrow{p_1p_2}\right\|_2}\right), & y_2-y_1\geqslant 0\\ -\operatorname{acos}\left(\dfrac{x_2-x_1}{\left\|\overrightarrow{p_1p_2}\right\|_2}\right), & y_2-y_1<0\end{cases},\ \theta\in(-\pi,\ \pi] \tag{3-14}$$

我们定义了光流的方向和二范数二维特征向量以区分静态对象表面和动态对象表面的光流，并认为动态对象光流与静态对象光流特征都服从高斯分布，所有样本集合则是高斯分布的线性组合，为高斯混合模型。对光流特征高斯混合模型进行聚类就是将光流特征样本划分为多个服从独立高斯分布函数的集合[82]。

本文中对图像 ORB 特征点光流进行观测，产生的观测数据为所有 ORB 光流特征集合，表示为：

$$\boldsymbol{X}=\{\boldsymbol{X}_1,\boldsymbol{X}_2,\cdots,\boldsymbol{X}_N\} \tag{3-15}$$

其中，$\boldsymbol{X}_i=[l\ \ \theta]^T$ 为光流的二维特征。光流二维特征的概率密度函数表示为：

$$P(\boldsymbol{X}|\boldsymbol{\Phi})=\frac{1}{2\pi|\boldsymbol{\Sigma}|^{1/2}}\exp[-\frac{1}{2}(\boldsymbol{X}-\boldsymbol{\mu})^T\boldsymbol{\Sigma}^{-1}(\boldsymbol{X}-\boldsymbol{\mu})] \tag{3-16}$$

其中，$\boldsymbol{\mu}$ 为各维特征均值，$\boldsymbol{\Sigma}$ 为特征协方差矩阵。

在本文中，高斯混合模型分量为 2，分别为动态对象表面光流高斯分量与静态对象表面光流高斯分量，因此，本文高斯混合模型为：

$$F(\boldsymbol{X}|\boldsymbol{\Theta})=\sum_{i=1}^{2}\alpha_i P(\boldsymbol{X}|\boldsymbol{\Phi}_i) \tag{3-17}$$

其中，α_i 为第 i 各高斯分量的先验概率，且 $\sum_{i=1}^{2}\alpha_i=1$，$\alpha_i\in[0,1]$，$P(\boldsymbol{X}|\boldsymbol{\Phi}_i)$ 为每个高斯分量的概率密度函数，如式(3-16)所示。

该高斯模型的参数即可用式(3-18)完全表示。

$$\boldsymbol{\Theta}_i=\{\alpha_i,\mu_i,\Sigma_i,\quad i=1,2\} \tag{3-18}$$

由于每个光流为单独采样，则每个光流样本为独立的，则样本集的联合条件密度函数为：

$$P(\boldsymbol{X}|\boldsymbol{\Theta})=\prod_{i=1}^{N}F(\boldsymbol{X}_i|\boldsymbol{\Theta})=\prod_{i=1}^{N}(\sum_{j=1}^{2}\alpha_j P(\boldsymbol{X}_i|\boldsymbol{\Phi}_j)) \tag{3-19}$$

通过使样本集的联合概率最大来对参数进行估计，从而计算出 GMM 的最佳分布模型，为简便计算，式(3-19)变换为该高斯混合模型的对数似然函数：

$$L(\boldsymbol{X}|\boldsymbol{\Theta})=\sum_{i=1}^{N}\log(\sum_{j=1}^{2}\alpha_j P(\boldsymbol{X}_i|\boldsymbol{\Phi}_j)) \tag{3-20}$$

至此，可采用 EM 算法对含有隐变量的 GMM 进行极大似然估计（Maximum Likelihood Estimation, MLE），本文隐变量即特征点所属类别。具体来说，该算法通过迭代的方式不断更新模型参数，以便最大化模型的似然函数。EM 算法的更新步骤通常包括以下 3 个部分：

(1) E-Step (Expectation step)：在当前模型参数的条件下，计算所有隐变量的期望。

(2) M-Step (Maximization step)：基于 E 步计算出的期望值，更新模型参数，以

最大化模型的似然函数。

(3)　重复 E 步和 M 步，直到模型参数趋近于某一固定值或达到算法设定的最大迭代次数。

令每个样本X_i对应的隐变量为z_{ik}, $k\in\{1,2\}$，表示样本X_i所属第k高斯分量，其取值为：

$$z_{ik}=\begin{cases}1, \text{样本}X_i\text{属于第}k\text{类高斯分量}\\ 0, \text{否则}\end{cases},\ i=1,2,\cdots,N;\ k=1,2 \tag{3-21}$$

定义后验概率$\Gamma(z_{ik})$表示样本X_i属于第k类的概率：

$$\begin{aligned}\Gamma(z_{ik})=P(z_{ik}=1|\boldsymbol{X})&=\frac{P(z_k=1)P(\boldsymbol{X}|z_k=1)}{\sum_{j=1}^{2}P(z_j=1)P(\boldsymbol{X}|z_j=1)}\\&=\frac{\alpha_k P(\boldsymbol{X}|\boldsymbol{\Phi}_k)}{\sum_{j=1}^{2}\alpha_j P(\boldsymbol{X}|\boldsymbol{\Phi}_j)}\end{aligned} \tag{3-22}$$

以下为 EM 算法在该问题具体的迭代步骤：

(1) E-Step

通过K-means算法[83]得到参数初始值$\boldsymbol{\Theta}_i^{init}, i\in\{1,2\}$，然后计算后验概率$\Gamma(z_{ik})$。

(2) M-Step

利用 MLE 求取参数$\boldsymbol{\Theta}_i, i\in\{1,2\}$估计值：

$$\begin{aligned}\hat{\boldsymbol{\Theta}}_{MLE}&=\arg\max_{\boldsymbol{\Theta}}\left(L(\boldsymbol{X}|\boldsymbol{\Theta})\right)\\&=arg\max_{\boldsymbol{\Theta}}\left(\sum_{i=1}^{N}\log\left(\sum_{j=1}^{2}\alpha_j P(\boldsymbol{X}_i|\boldsymbol{\Phi}_j)\right)\right)\end{aligned} \tag{3-23}$$

然后，用式(3-23)计算的估计值更新第k个高斯分量的参数：

$$\begin{cases}\mu_k^{t+1}=\dfrac{1}{P_k}\sum_{i=1}^{2}\Gamma(z_{ik})x_i\\ \boldsymbol{\Sigma}_k^{t+1}=\dfrac{1}{P_k}\sum_{i=1}^{2}\Gamma(z_{ik})(x_i-\mu_k^{t+1})(x_i-\mu_k^{t+1})^T\\ \alpha_k^{t+1}=\dfrac{P_k}{2}\end{cases} \tag{3-24}$$

其中，t为迭代次数，$P_k=\sum_{i=1}^{2}\Gamma(z_{ik})$。

M-Step 结束后，返回执行 E-Step，直到满足本文中设定的停止条件如式(3-25)所示，即对数似然函数变化小于 0.1 或迭代次数大于等于 20 次。

$$|L_{t+1}(\boldsymbol{X}|\boldsymbol{\Theta}) - L_t(\boldsymbol{X}|\boldsymbol{\Theta})| < 0.1 \ \vee \ t \geqslant 20 \tag{3-25}$$

3.3.3 基于 GMM 的静态特征点筛选

为了对潜在运动对象检测框内的静态特征点进行筛选，设计了基于 GMM 的静态特征点筛选算法，如算法 1 所示。

算法 1 基于GMM的静态特征点筛选算法

输入： 动态对象检测框$Boxes$, 特征点$mvKey$, 特征金字塔索引$Level$
输出： 静态特征点$KeyPointWithoutDynamic$

```
q ← Queue(4)                                          ▷ 初始化大小为4的FIFO
tmpKey ← mvKey
for each i ∈ [0, Level − 1] do                        ▷ 步骤1
    K_i ← mvKey[i]
    for each keypoint ∈ K_i do
        if keypoint.pos ∈ Boxes then
            K_i.remove(keypoint)
        end if
    end for
end for

if q.isfull() then
    q.push(tmpKey)
    curKey ← q.front()
    trackingPoints ← calcOpticalFlowPyrLK(curKey)     ▷ 步骤2
    DirAndLegth ← calOpticalFlowDirAndLength(trackingPoints)
    expectationMaximizationAlgorithm(DirAndLegth)     ▷ 步骤3
    flag ← getCorrectClass()
    for each point ∈ trackingPoints do
        if flag == point.class then
            addKeyPointToBox(mvKey)                   ▷ 步骤4
        end if
    end for
else
    q.push(tmpKey)
end if
KeyPointWithoutDynamic ← mvKey
return KeyPointWithoutDynamic
```

其中，算法输入$Boxes$为潜在运动对象检测框，$mvKey$为 ORB 特征点，$level$为提取的 ORB 特征所在特征金字塔的层索引，算法输出$KeyPointWithoutDynamic$为不含动态特征点的 ORB 特征点。

该算法可以大体分为 4 个步骤。

(1) 潜在运动对象区域原始 ORB 特征移除（步骤 1）

在输入新的一帧图像时，对该图像中潜在动态对象区域进行特征点删除。由于 ORB 特征提取采用了特征金字塔以保证 ORB 的尺度不变性，而目标检测只在原始图像分辨率进行，在判断特征点是否属于该区域时，需要将ORB特征点每一层特征点恢复到底层：

$$p_l = p_i \times s_i \tag{3-26}$$

其中，p_l 为原始分辨率下的特征点位置，p_i 为特征金字塔第 i 层的特征点位置，s_i 为第 i 层特征金字塔与原始图像的缩放尺度。剔除潜在对象区域特征点后的特征点如图 3-9 所示。

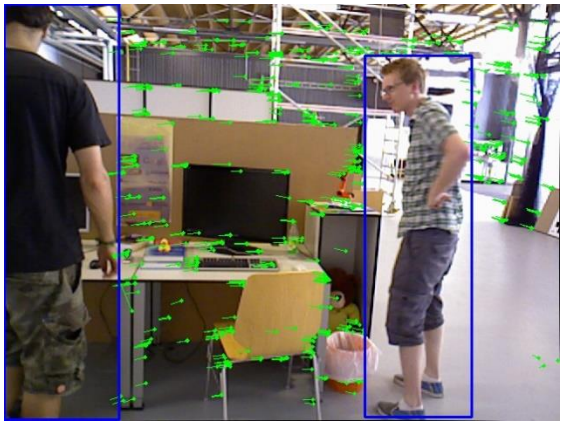


图 3-9 剔除潜在运动对象区域后的特征点

(2) 计算光流特征（步骤 2）

在算法开始时定义了大小为 4 的滑动窗口在图像序列上滑动，滑动窗口在实现上采用先进先出（FIFO）数据结构。在队列未满时，对每一帧图像 ORB 特征入队，当队列满时，在每次新的一帧图像到达时，先队尾图像 ORB 特征出队，然后将对该图像原始 ORB 特征入队，以方便后续恢复静态特征，如图 3-10 所示。

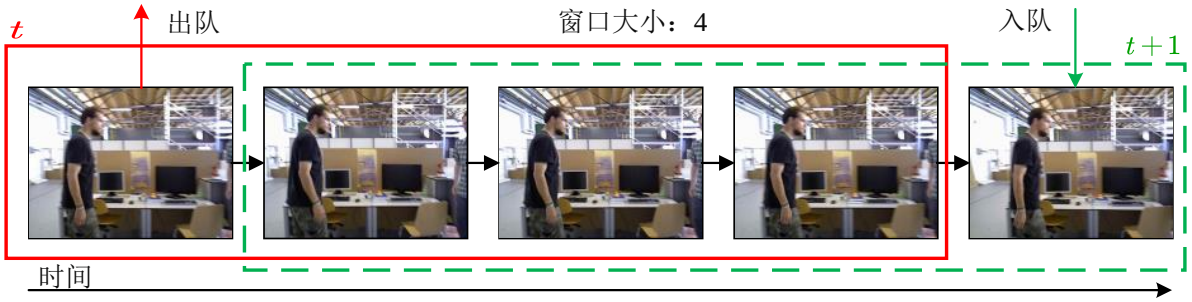


图 3-10 滑动窗口

当滑动窗口满时，则以队首为当前帧，队尾为下一帧，对整幅图像的 ORB 特征实施 LK 光流，并利用式(3-13)与式(3-14)对成功跟踪的光流计算光流二范数与方向。值得注意的是，并非所有特征都能成功被跟踪，因此可以认为在进行该步骤时，对不稳定的特征进行了一次过滤。后续，未被跟踪的特征点将不会参与到静态特征与动态特征的分类中，自然也就不会被重新添加到算法输出的静态特征点集合。

(3) 利用 EM 算法对光流特征进行聚类（步骤 3）

为了提高估计的准确性，以整幅图像的特征点光流二范数与方向构建高斯混合模型，然后利用 EM 算法对该模型的参数进行极大似然估计，最后根据特征点隐变量所属类别的后验概率较大者作为特征点的标签类别，具体算法见 3.3.2 节。因为 EM 算法

本质上为聚类算法，仅能根据特征点光流的概率分布将特征点划分为两个不同的类别，而不能直接确定某个特征点属于静态特征还是动态特征。为了得到每个特征点的确切所属类别，我们设计了以下算法 2 进行静态特征点筛选。

算法 2 静态特征点标签识别算法

输入： 被跟踪的特征点$trackingPoints$, 潜在运动区域$Boxes$, EM算法输出的光流特征类别$Labels \in \{0,1\}$

输出： 静态特征点类别$staticPointClass \in \{0,1\}$

```
outBoxesPointNum←0
vector classCount(2)
for each keypoint∈trackingPoints do
    if keypoint.pos∉Boxes then
        outBoxesPointNum←outBoxesPointNum + 1
        classCount[Labels[i]]←classCount[Labels[i]] + 1
    end if
end for
staticPointClass←MaxElement(classCount)
return staticPointClass
```

其中，$trackingPoints$ 为 LK 光流跟踪到的 ORB 特征，$Boxes$ 为潜在动态区域，$Labels$ 为EM算法输出的每个被跟踪的光流特征类别，$staticPointClass$ 为静态特征点类别。该算法对潜在动态区域外的光流特征类别进行统计，认为数量多者为静态特征。具体来说，若潜在动态区域外$Labels=0$的特征数量为 50 个，而$Labels=1$的数量为 20 个，则认为$Labels=0$为静态特征点的类别。

(4) 潜在运动对象区域静态特征点恢复（步骤 4）

最后遍历潜在运动对象区域的 ORB 特征点，若其属于静态特征点则重新添加入 ORB 特征集合，如图 3-11 所示。图中，蓝色矩形框为 PicoDet 输出的潜在运动对象区域，绿色线条为算法输出的静态特征光流，红色线条为算法输出的动态特征光流。

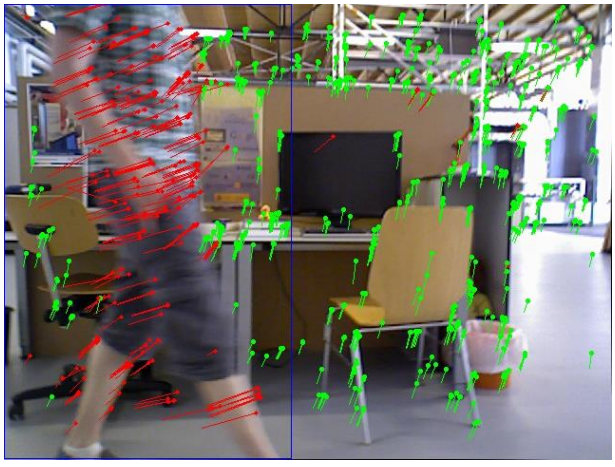


图 3-11 静态特征点与动态特征点示意图

3.4 动态环境下的 RGBD SLAM 系统构建

本文将静态特征点筛选算法集成到 ORB-SLAM2[17]中，整体系统框架如图 3-12 所

示。其中绿色部分为本章主要工作。系统主要由五个线程组成，分别为潜在运动区域检测线程、修改之后的跟踪线程、局部建图线程、闭环校正线程和全局 BA 线程。

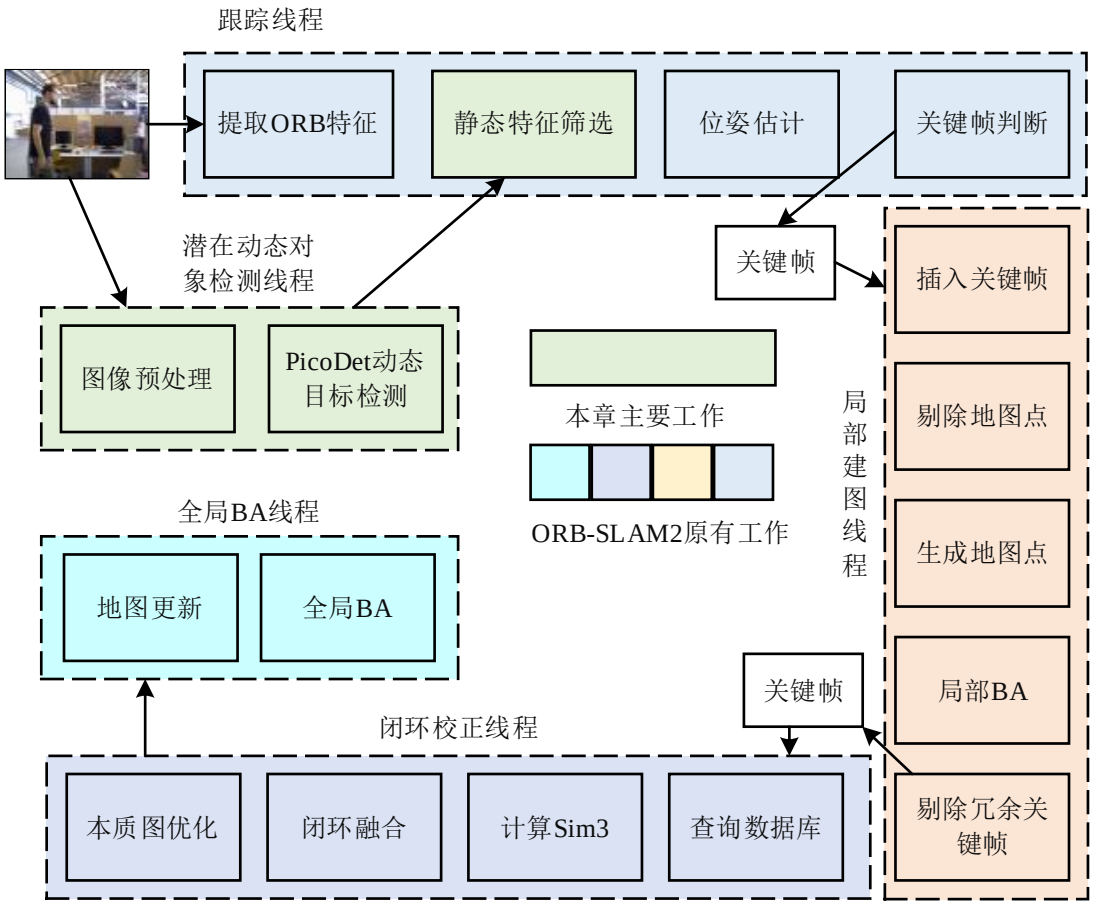


图 3-12 SLAM 系统框图

每个线程主要完成的工作为：

(1) 潜在动态对象检测线程

该线程运行超轻量目标检测算法 PicoDet 对预先定义的潜在动态对象进行检测，输入为 RGB 图像，输出为潜在动态对象的位置。

(2) 跟踪线程

首先提取 ORB 特征，其次使用本文静态特征点筛选算法执行静态特征点筛选，并完成特征点匹配，然后使用 PnP 方法进行位姿估计，最后筛选关键帧并传给局部建图线程。

(3) 局部建图线程

该线程处理新的关键帧，用于进一步优化位姿和地图，在此过程中还需要对地图点进行筛选和添加，最后删除冗余的关键帧以减轻后续线程的压力。

(4) 闭环校正线程

该线程用于检测相机是否移动到了之前的位置，通过回环的修正来减小累积误差。回环检测成功后先进行回环修正，最后触发全局 BA。

(5) 全局 BA 线程

该线程不与其他线程同步运行，而是由闭环校正线程触发。该线程对全局地图和轨迹进行优化。

3.5 实验与分析

为了对本章算法进行验证，首先对本章算法的潜在动态对象筛选模块进行了实验和定性分析，然后可视化分析了静态特征点筛选算法，最后在 TUM RGBD 公开数据集对本章算法整体进行了实验，并与经典的 ORB-SLAM2[17]、DS-SLAM[28]进行了对比，以测试该算法在复杂动态场景中的鲁棒性和定位精度。

TUM RGBD 数据集是 Schubert 等人[64]提出的用于评估基于 RGBD 相机的 SLAM 系统性能指标的数据集。该数据集采用微软 Kinect RGBD 相机进行采集，相机帧率设定为30FPS，分辨率640×480。同时，该数据集利用运动捕获系统采集了相机真实的运动轨迹，为基于 RGBD 相机的 SLAM 算法提供了统一的评估基准。该数据集下的每个序列都包含了由 Kinect 同步采集的原始 RGBD 图像和真实相机位姿。本文主要针对动态场景进行研究，选择了该数据集中针对动态场景的一个子集，包含多个高动态场景序列（f3_walking_*）和多个低动态场景序列（f3_sitting_*）。该子集采用手持相机的方式进行采集，其中三个采集序列如图 3-13 所示，第一行为 RGB 数据，第二行为深度数据。

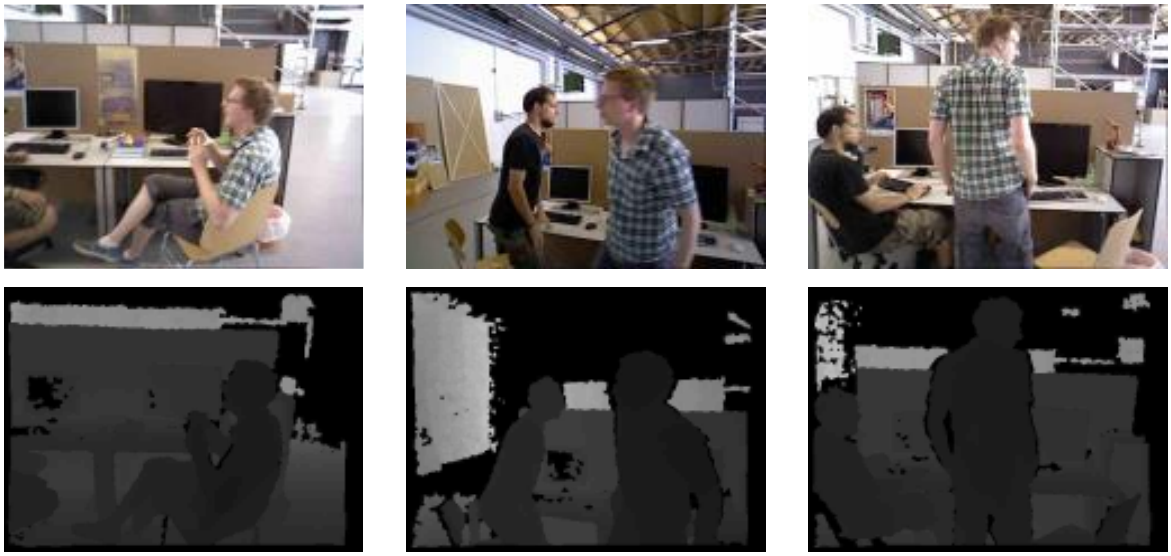


图 3-13 TUM RGBD 动态场景数据集示例

3.5.1 超轻量目标检测算法验证

超轻量目标检测算法能够检测的类型较多，在 TUM RGBD 数据集中，存在的潜在动态对象仅有人体，因此将目标检测算法调整为只对人体进行检测。图 3-14 给出了超轻量目标检测算法 PicoDet 在 TUM RGBD 数据集部分序列上的检测结果。各子图表

示该组 RGB 图片所在序列，黄色数字表示该图片的在所在序列的编号，蓝色方框表示 PicoDet 检测到的潜在动态对象。

(a) walking_xyz

(b) walking_static

(c) sitting_rpy

(d) sitting_halfsphere

图 3-14 PicoDet 在 TUM RGBD 数据集部分序列检测检测

从 walking_xyz 序列第 100 帧、walking_static 序列第 150 帧可知，PicoDet 目标检测算法能够实现对完整人体的准确检测；从 walking_xyz 序列第 350 帧、sitting_halfsphere 序列第 400 帧可知，在人体部分区域出现在图像中且遮挡图像大部分区域时，PicoDet 算法能够实现对人体的准确检测；从 walking_static 序列第 150 帧、sitting_halfsphere 序列第 100 帧可知，PicoDet 算法能够检测不同姿态下的人体；从 sitting_rpy 序列的第 400 帧与第 450 帧可知，PicoDet 算法对相机旋转依然鲁棒，在相

机大角度旋转时仍然能够对人体进行有效检测；在 sitting_rpy 序列第 300 帧，PicoDet 检测出了右边人物头部，而左边人物则没有成功检测，当人体极少部分出现在图像中时，PicoDet算法可能存在漏检，但这并不影响本章算法的有效性，因为潜在动态对象若只出现小部分在图像中，那这部分将只存在极少数甚至没有动态特征点，对本章算法造成的影响也不必考虑。

为了验证 PicoDet 算法的实时性，对 PicoDet 全精度模型进行量化，然后利用腾讯 NCNN 神经网络前向计算框架将模型部署在多种平台使用 CPU 进行推理，在 TUM RGBD 数据集各序列进行测试，计算得到各平台的平均帧率见表 3-1。在 PC（AMD Ryzen7 R4800H）推理，帧率为 55FPS，在机载计算机（CPU 为 I7-8565U）进行推理，帧率达 40FPS，在 Nvidia Jetson TX2 嵌入式板卡（双核 Denver 2 64-bit CPU + 四核 ARM A57）进行推理，帧率为 21FPS，满足实时性需要。

表 3-1 PicoDet 算法在各平台 CPU 上推理帧率

平台	AMD Ryzen7 R4800H	Intel I7-8565U	Nvidia Jetson TX2
帧率	55	40	21

从以上分析可知，本章 3.2 小节的 PicoDet 算法能够在 TUM RGBD 数据集中大部分场景下完成对潜在动态对象的检测，且仅使用 CPU 推理时满足实时性要求，满足本章算法对潜在动态对象进行检测的需要。PicoDet能够高质量完成潜在动态对象的检测，得益于该算法改进后的骨干网络 ES-Net、重新轻量化设计后的颈部以及先进的模型训练策略，这些优化提高了 PicoDet 的特征提取能力和效率。

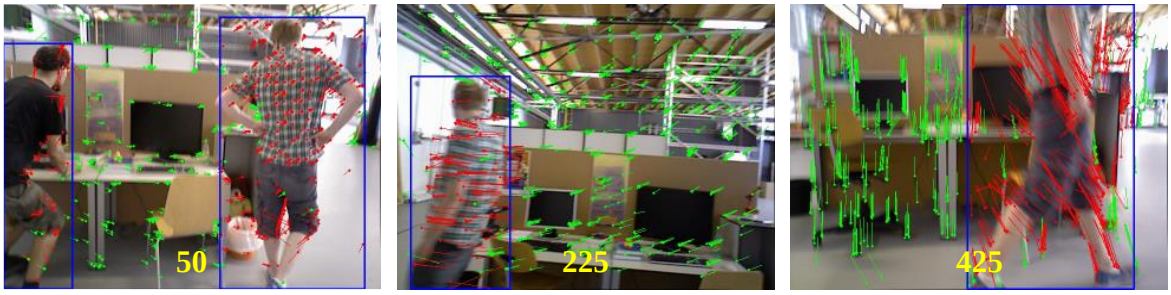
3.5.2 静态特征点筛选算法验证

图 3-15 给出了本章静态特征点筛选算法在 TUM RGBD 数据集部分序列的典型筛选结果。图中各子图表示该组RGB图片所在序列，黄色数字表示该图片的在所在序列的编号，蓝色方框表示 PicoDet 检测到的潜在运动对象，绿色线条和点表示静态特征点筛选算法筛选出的通过光流追踪到的静态特征点，红色线条和点表示潜在运动区域的通过光流追踪到的动态特征点。检测框外的绿色线条长度可以一定程度上反映相机的运动速度，框内人体身上的红色线条长度可以一定程度反映人体的运动速度。

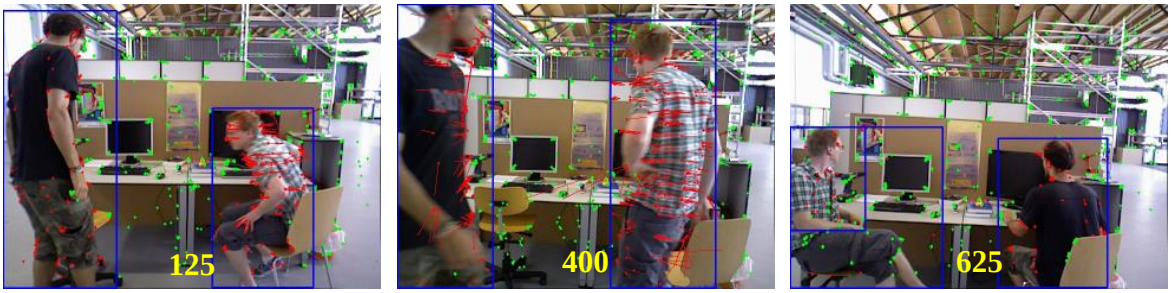
从 walking_xyz 序列第 50 帧相机和人体都运动缓慢，第 225 帧相机运动缓慢人体快速移动、第 425 帧相机和人体都快速移动时，静态特征点筛选算法可以检测到人体的运动，同时保留检测框内少许静态特征点。从 walking_static 序列来看，当相机缓慢移动，人体遮挡大量静态场景时，筛选算法能够有效区分静态特征点和动态特征点。在 walking_static 序列第 625 帧、sitting_static 序列的第 400 帧与第 625 帧，人体部分存在运动时，静态特征点筛选算法能保留检测框内电脑和桌子等静态物体特征点的同时保留人体静止部位的特征点。从 sitting_rpy 序列的第 25 帧和第 600 帧可知，相机大角度旋转时，静态特征点筛选算法同样能够筛选出人体上的动态特征点，但检测框内少

许静态物体上的特征点被误检为动态特征点，从 sitting_rpy 序列的第 350 帧可知，在相机高速运动时导致的运动模糊场景，静态特征点筛选算法也能够准确筛选出场景中的静态特征点与动态特征点。

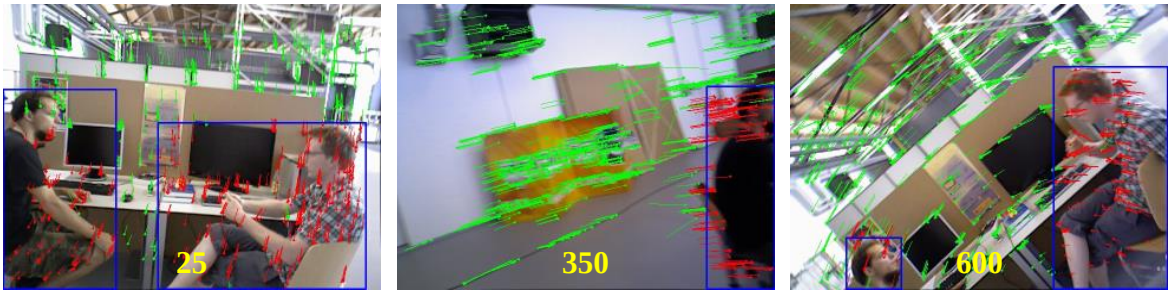
从以上分析可知，本章 3.3 小节的静态特征点筛选算法在 TUM RGBD 数据集的多种场景中能够较为准确的区分出潜在运动区域内的特征点类别。由于该算法在设计时利用了光流的方向，在相机存在大角度旋转时的区域性光流方向一致将会给算法带来挑战，导致算法对特征点存在少量的误筛选。即便如此，该算法在大部分场景中的鲁棒性仍能够满足本章 SLAM 算法的需要。



(a) walking_xyz



(b) walking_static



(c) sitting_rpy

(d) sitting_static

图 3-15 静态特征点筛选算法在 TUM RGBD 数据集部分序列筛选效果

3.5.3 TUM RGBD 数据集实验

将本章算法、ORB-SLAM2 以及 DS-SLAM 在 TUM RGBD 数据集上进行对比实验，使用多个评价指标对三种算法进行定量评估，以验证本文算法在复杂动态环境中的鲁棒性。本节实验的评价指标为绝对轨迹误差（ATE）和相对位姿误差（RPE），采用均方根误差（RMSE）、均值（Mean）、中位数（Median）、标准差（STD）进行统计。表 3-2 和表 3-3 给出了三种算法在 TUM RGBD 数据集不同序列中的 ATE 和 RPE，表中“**加粗**”字体表示运行于其中一个序列的算法（行）在当前评价指标（列）上结果最优，表格带“↓”的四列表示本章算法相较于其他两种算法的误差变化。

表 3-2 算法绝对轨迹误差（ATE）对比

序列	算法	均方根误差	均值	中位数	标准差	均方根误差↓	均值↓	中位数↓	标准差↓
sitting_rpy	ORB-SLAM2	0.0222	0.0172	0.0130	0.0139	30.8%	28.6%	21.5%	34.3%
	DS-SLAM	0.0197	0.0143	**0.0101**	0.0136	22.2%	13.8%	-0.9%	32.6%
	本章算法	**0.0153**	**0.0123**	0.0102	**0.0091**	–	–	–	–
sitting_static	ORB-SLAM2	0.0082	0.0072	0.0065	0.0038	13.9%	12.7%	10.9%	18.0%
	DS-SLAM	**0.0066**	**0.0057**	**0.0050**	0.0033	-6.4%	-10.4%	-15.4%	6.2%
	本章算法	0.0070	0.0063	0.0058	**0.0031**	–	–	–	–
sitting_xyz	ORB-SLAM2	**0.0093**	**0.0080**	**0.0073**	**0.0048**	-17.6%	-18.7%	-16.9%	-14.4%
	DS-SLAM	0.0103	0.0091	0.0083	0.0049	-6.1%	-4.5%	-3.4%	-11.1%
	本章算法	0.0109	0.0095	0.0085	0.0055	–	–	–	–
walking_hs	ORB-SLAM2	0.5847	0.5028	0.4300	0.2986	95.0%	95.0%	95.1%	94.9%
	DS-SLAM	0.0328	0.0279	0.0226	0.0174	10.1%	9.0%	6.4%	13.1%
	本章算法	**0.0295**	**0.0253**	**0.0212**	**0.0151**	–	–	–	–
walking_rpy	ORB-SLAM2	0.7093	0.5777	0.3910	0.4116	96.0%	96.4%	95.8%	95.4%
	DS-SLAM	0.3669	0.3242	0.2628	0.1719	92.3%	93.5%	93.8%	88.9%
	本章算法	**0.0284**	**0.0210**	**0.0163**	**0.0190**	–	–	–	–
walking_static	ORB-SLAM2	0.1641	0.1505	0.1738	0.0654	95.4%	95.8%	96.7%	93.5%
	DS-SLAM	**0.0069**	**0.0062**	0.0062	**0.0029**	-10.9%	-1.5%	6.9%	-46.7%
	本章算法	0.0076	0.0063	**0.0057**	0.0042	–	–	–	–
walking_xyz	ORB-SLAM2	0.7283	0.6297	0.5476	0.3658	97.8%	97.9%	97.8%	97.8%
	DS-SLAM	0.2288	0.2139	0.2188	0.0814	93.2%	93.7%	94.6%	90.3%
	本章算法	**0.0157**	**0.0135**	**0.0119**	**0.0079**	–	–	–	–

从表 3-2 可知，本章算法相比 ORB-SLAM2，在大部分序列取得了更低的 ATE；相比 DS-SLAM，本章算法在高动态场景具有显著优势，在低动态场景定位误差相当。特别地，在 walking_hs（人体移动、相机半球运动）、walking_rpy（人体移动、相机小平移大旋转）、walking_xyz（人体移动、相机平移）等高动态场景，本章算法相比 ORB-SLAM2 和 DS-SLAM 有更低的 ATE，均方根误差最低仅为$0.0157m$，相较于 ORB-SLAM2 降低97.8%，相较于 DS-SLAM 降低93.2%，其余指标也均有大幅降低。

在所有高动态场景（walking_*）中，本章算法的 ATE 均方根误差相比 ORB-SLAM2 平均降低96.0%，相比 DS-SLAM 平均降低了46.2%。在 sitting_rpy（人体坐立，相机小平移大旋转）和 sitting_static（人体坐立、相机基本静止）低动态场景序列，本章算法相较于 ORB-SLAM2 有着更低的 ATE，均方根误差分别降低了30.8%和13.9%；相较于 DS-SLAM，两种算法在不同评价指标各有优势，总体差距不大。在序列 sitting_xyz，ORB-SLAM2 相比本章算法和 DS-SLAM 在所有评价指标中都取得了最低的 ATE，但差距也并不明显，可能的原因为 ORB-SLAM2 中使用的随机采样一致性算法能够过滤低动态场景中的部分外点并保留了尽可能多的静态特征点，使得在动态对象运动并不显著的场景中取得了更优的结果。从以上分析可以发现，场景越复杂、动态对象移动幅度越大，相比其余两种算法本章算法所取得的 ATE 越低，定位精度越高。

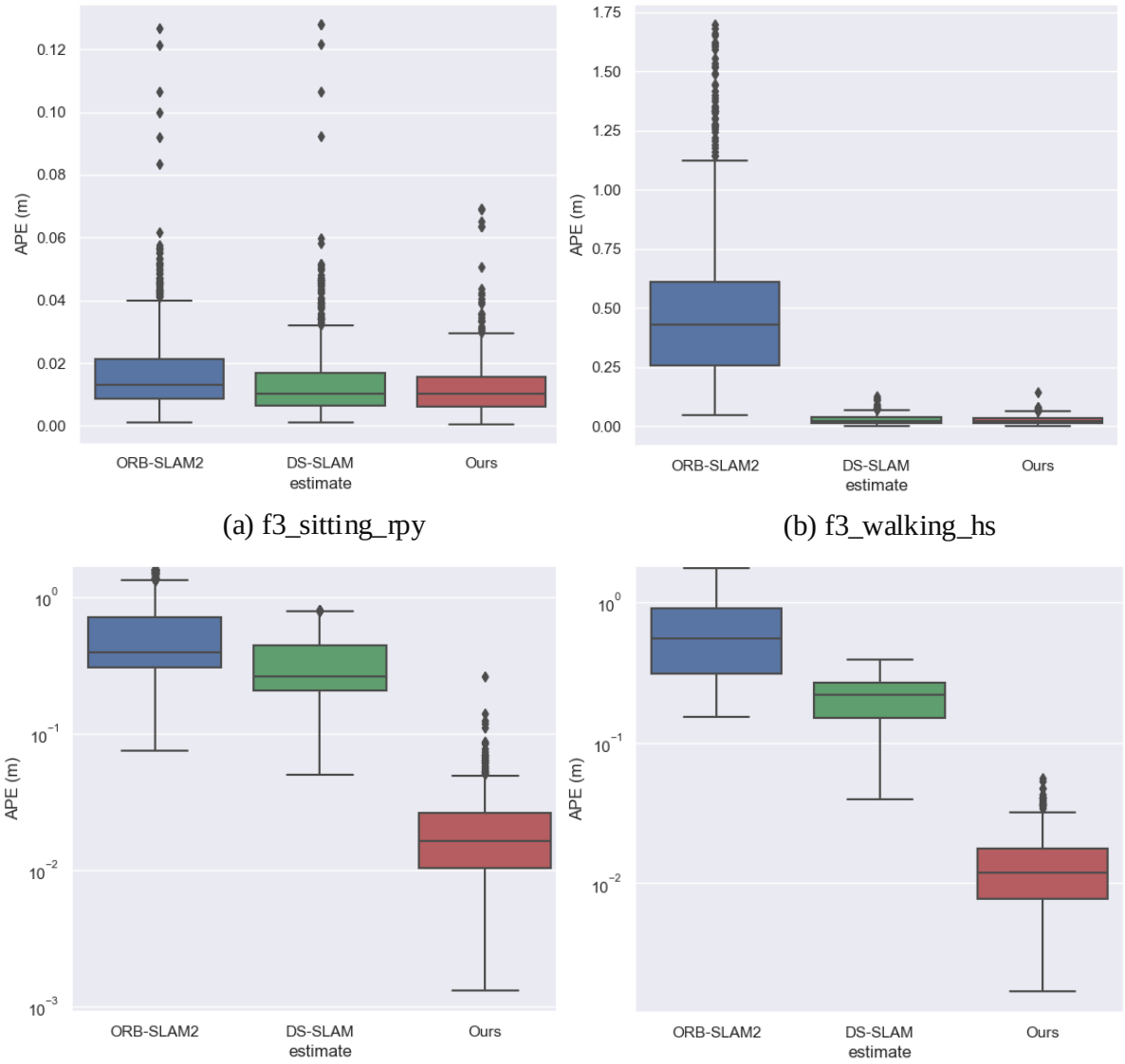
表 3-3 算法相对位姿误差（RPE）对比

序列	算法	均方根误差	均值	中位数	标准差	均方根误差↓	均值↓	中位数↓	标准差↓
sitting_rpy	ORB-SLAM2	**0.0136**	0.0100	0.0081	0.0091	-2.6%	0.8%	4.9%	-6.7%
	DS-SLAM	0.0140	0.0106	0.0079	**0.0092**	0.6%	5.8%	3.2%	-6.0%
	本章算法	0.0139	**0.0100**	**0.0077**	0.0097	–	–	–	–
sitting_static	ORB-SLAM2	0.0049	0.0042	0.0037	**0.0025**	6.1%	9.6%	16.8%	-3.2%
	DS-SLAM	0.0063	0.0055	0.0050	0.0030	26.2%	30.7%	38.1%	12.5%
	本章算法	**0.0046**	**0.0038**	**0.0031**	0.0026	–	–	–	–
sitting_xyz	ORB-SLAM2	0.0084	**0.0072**	**0.0063**	**0.0044**	-1.8%	-1.0%	-1.1%	-4.0%
	DS-SLAM	0.0105	0.0088	0.0077	0.0056	18.6%	18.2%	17.2%	19.8%
	本章算法	**0.0085**	0.0072	0.0064	0.0045	–	–	–	–
walking_hs	ORB-SLAM2	0.0463	0.0182	0.0131	0.0426	70.0%	41.1%	35.0%	79.3%
	DS-SLAM	0.0148	0.0116	0.0094	0.0093	6.5%	7.2%	9.6%	5.4%
	本章算法	**0.0139**	**0.0107**	**0.0085**	**0.0088**	–	–	–	–
walking_rpy	ORB-SLAM2	0.0317	0.0233	0.0173	0.0214	25.4%	26.7%	25.3%	23.8%
	DS-SLAM	0.0252	**0.0163**	**0.0114**	0.0193	6.4%	-5.1%	-12.8%	15.5%
	本章算法	**0.0236**	0.0171	0.0129	**0.0163**	–	–	–	–
walking_static	ORB-SLAM2	0.0170	0.0111	0.0067	0.0129	69.6%	60.1%	42.1%	79.2%
	DS-SLAM	0.0071	0.0057	0.0049	0.0043	27.6%	22.5%	20.1%	37.4%
	本章算法	**0.0052**	**0.0044**	**0.0039**	**0.0027**	–	–	–	–
walking_xyz	ORB-SLAM2	0.0247	0.0205	0.0170	0.0139	48.7%	49.4%	50.9%	47.1%
	DS-SLAM	0.0151	0.0114	0.0085	0.0099	16.0%	9.0%	1.9%	26.2%
	本章算法	**0.0127**	**0.0104**	**0.0084**	**0.0073**	–	–	–	–

从表 3-3 可知，在高动态场景序列（walking_*）中，本章算法相较于 ORB-SLAM2 和 DS-SLAM 有着更低的 RPE，均方根误差相较于 ORB-SLAM2 平均降低了53.4%，相较于 DS-SLAM 平均降低了14.1%。在部分低动态场景序列（sitting_rpy、sitting_xyz）中本章算法 RPE 在部分指标误差略大于 ORB-SLAM2，主要原因同样为 ORB-SLAM2 的随机一致性算法起到了剔除外点的作用。整体来说，本章算法的在动

态场景的相对位姿精度优于 ORB-SLAM2 和 DS-SLAM，具有更强抑制漂移的能力。

为了更为直观的展示本章算法的定位精度和稳定性，绘制了本章算法与其余两种算法其中四个序列的箱线图，图 3-16 所示为绝对轨迹误差箱型图。由于本文算法的绝对轨迹误差较小，为了更清晰的展示三种算法的误差分布，图 3-16(c)和(d)将纵坐标轴坐标映射为指数坐标。



(a) f3_sitting_rpy

(b) f3_walking_hs

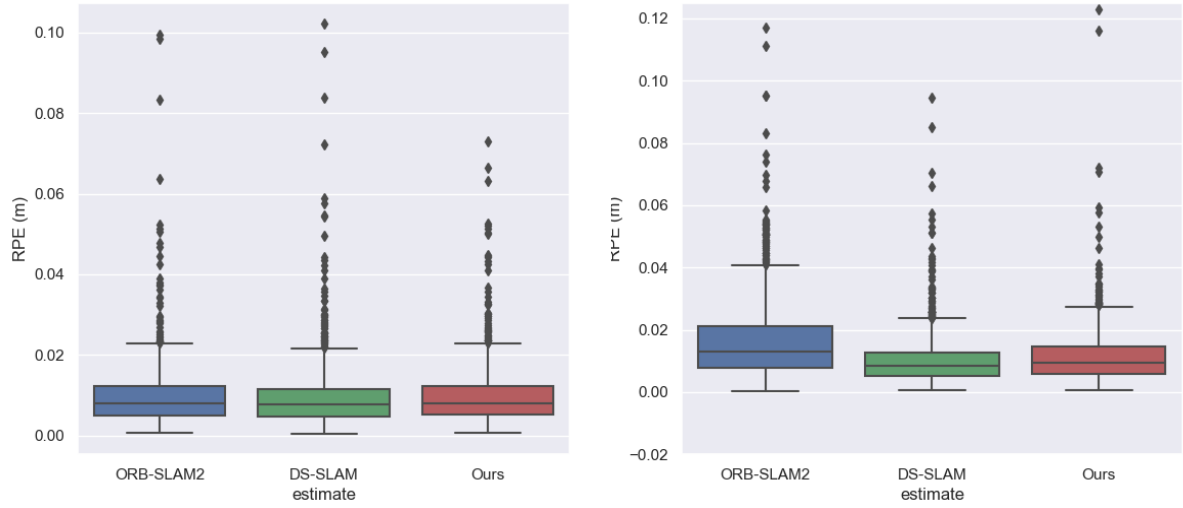
(c) f3_walking_rpy

(d) f3_walking_xyz

图 3-16 绝对轨迹误差箱线图

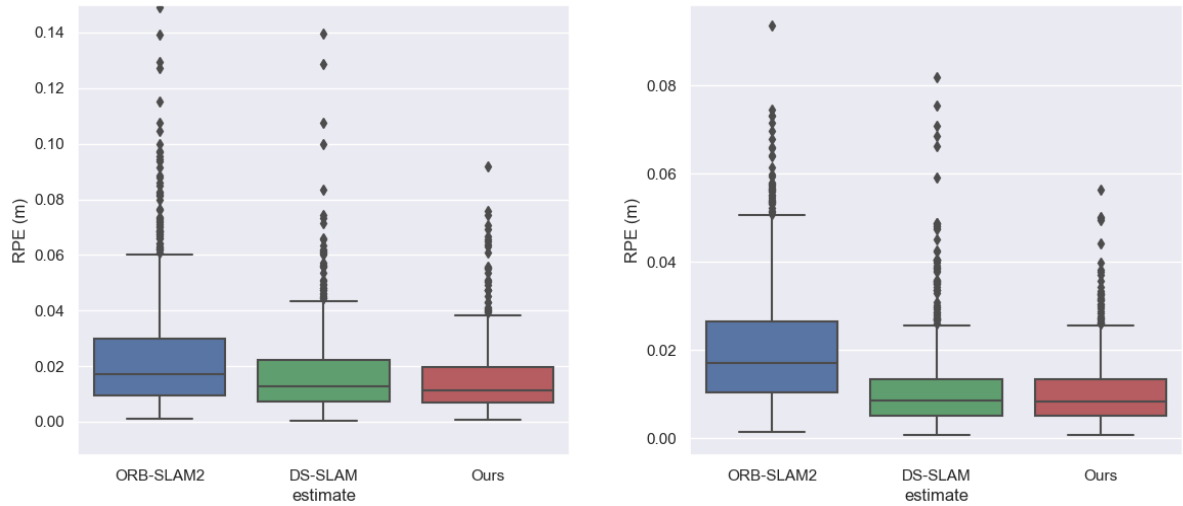
由图 3-16 可知，本章算法的 ATE 平均误差相比其他两种算法更低。特别地，从图 3-16(c)和(d)高动态场景序列的箱线图可以发现，本章算法 ATE 误差分布比其余两种算法低 1 个数量级。同时，本章算法 ATE 误差分布比其余两种算法更为集中，说明本章算法的绝对轨迹精度在高动态场景比其余两种算法更稳定。

图 3-17 展示了本章算法与其余两种算法的相对位姿误差箱线图。



(a) f3_sitting_rpy

(b) f3_walking_hs



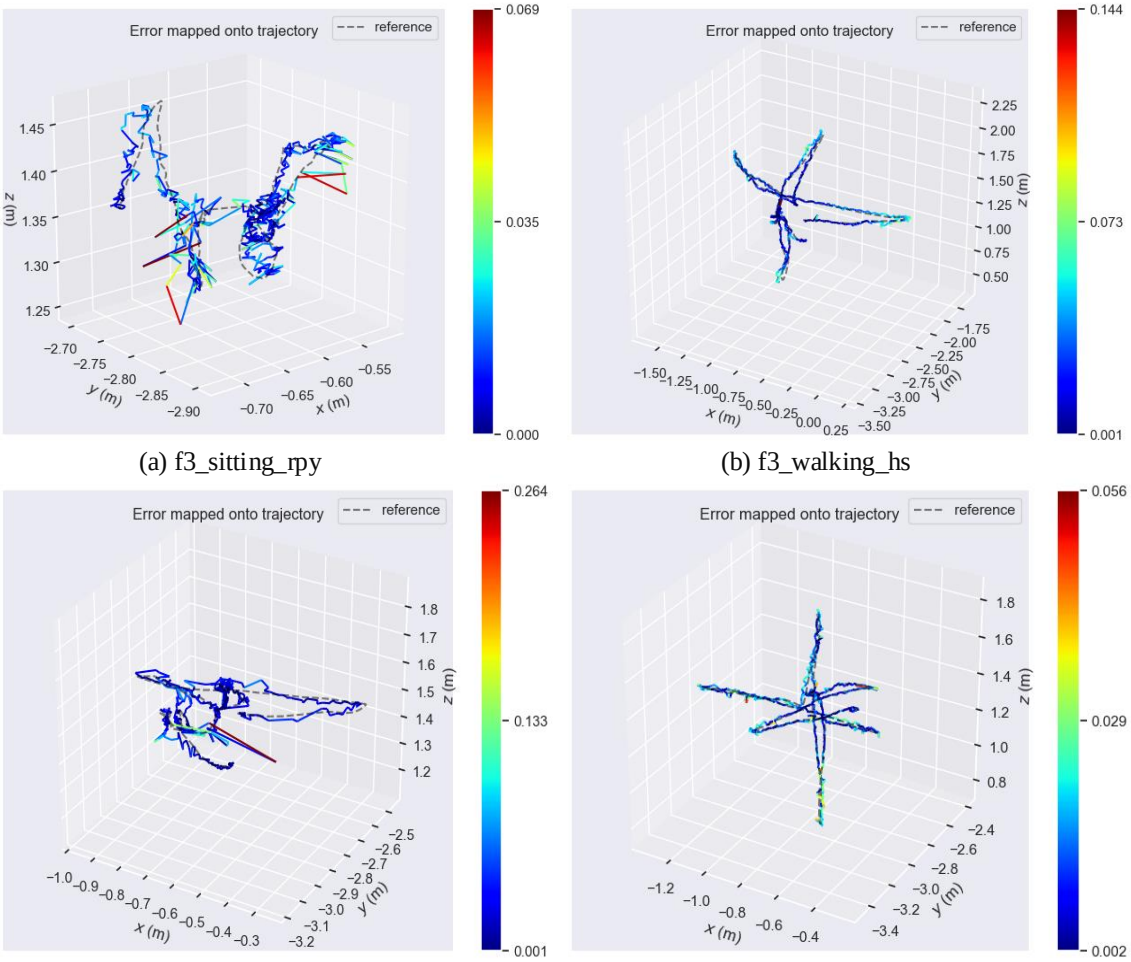
(c) f3_walking_rpy

(d) f3_walking_xyz

图 3-17 相对位姿误差箱线图

从图 3-17 可知，本章算法的 RPE 误差分布与 DS-SLAM 相近，但离群点相对较少，ORB-SLAM2 的 RPE 比其余两种算法更为分散。说明本章算法的 RPE 在高动态场景相比 ORB-SLAM2 更低，比 DS-SLAM 更具有稳定性。

图 3-18 给出了本章算法得到的轨迹在其中四个序列与真实轨迹的可视化效果图。真实轨迹和本章算法估计的轨迹分别用虚线和实线表示，渐变实线表示同一时间戳的真实轨迹与估计轨迹的误差，从蓝色到红色表示误差从小变大。图 3-18(a)展示了本章算法在 f3_sitting_rpy 序列的轨迹与真实轨迹的差异，可以看到，线条多处出现红色线段，并与真实轨迹相差较大。这是因为该序列存在大面积人体移动和运动模糊，同时判定为静态特征的数量偏少，以至于导致算法无法正常跟踪。即便如此，本章算法相比其余两种算法在 f3_sitting_rpy 序列也取得了更好的定位效果。在展示的其余三组序列中，本章算法都取得了较好的定位效果。



(a) f3_sitting_rpy　　(b) f3_walking_hs

(c) f3_walking_rpy　　(d) f3_walking_xyz

图 3-18 本章算法轨迹与真实轨迹比较图

为了验证本章算法的实时性，分别在 PC（AMD Ryzen7 R4800H）、机载计算机（CPU 为 I7-8565U），Nvidia Jetson TX2 嵌入式板卡（双核 Denver 2 64-bit CPU + 四核 ARM A57）进行实验，对 TUM RGBD 数据集 7 个序列运行测试，计算得到每种算法在各平台的平均帧率如表 3-4 所示，条形图如图 3-19 所示。表中可见，ORB-SLAM2 算法能够在各个平台正常运行，且帧率最高；DS-SLAM 算法由于需要使用 GPU 进行语义分割，因此不能在没有独立 GPU 的机载计算机上正常运行，并且在测试的所有平台上帧率最低；本章算法虽然需要进行潜在动态对象检测，但能够在 CPU 上快速推理，因此能够在测试的各类平台正常运行，本章算法在机载计算机上平均帧率为 20 帧，满足无人机自主飞行需求。

表 3-4 三种算法在各平台平均帧率对比

平台	ORB-SLAM2	DS-SLAM	本章算法
PC	40	21	25
机载计算机	31	—	20
Jetson TX2	15	9	12

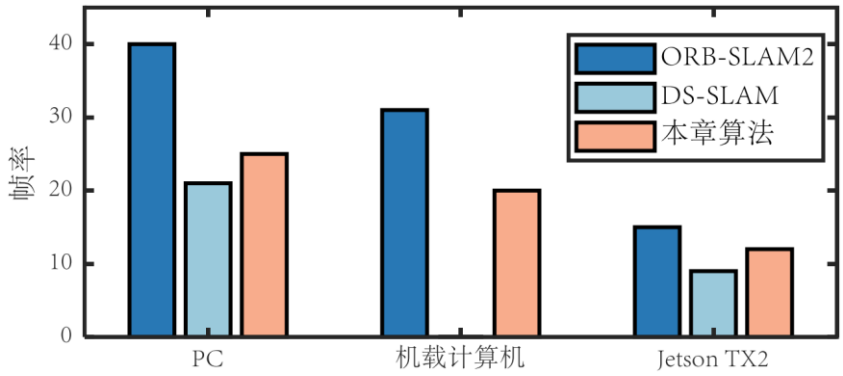


图 3-19 三种算法在各平台平均帧率对比条形图

3.6　本章小结

本章对无人机自主飞行的关键技术之一——SLAM 算法展开研究。针对传统视觉 SLAM 在复杂动态场景中定位精度较差的问题，本章提出了一种基于静态特征点筛选的动态环境 RGBD SLAM 算法。首先，利用超轻量目标检测算法 PicoDet 对动态场景中的潜在动态对象进行预筛选。其次，将特征点光流的二范数与方向构建为二维特征向量，以构建高斯混合模型，并使用 EM 算法对该模型进行聚类。然后，利用高斯混合模型聚类结果保留潜在动态区域的静态特征点，以提高位姿估计精度。最后，本章将以上算法集成到 ORB-SLAM2 前端中，并在 TUM RGBD 公开数据集进行实验。结果表明，在该数据集的高动态场景序列中，本章算法的 ATE 均方根误差相比 ORB-SLAM2 平均降低96.0%，相比 DS-SLAM 平均降低了46.2%；本章算法的 RPE 均方根误差相较于 ORB-SLAM2 平均降低了53.4%，相较于 DS-SLAM 平均降低了14.1%。本章算法能够有效提升 SLAM 系统在复杂动态环境中的定位精度和鲁棒性，同时无需 GPU 参与，能够在无人机的机载计算机上取得较高的帧率。

4　基于样条曲线的多旋翼无人机局部运动规划

4.1　引言

运动规划同样是无人机自主飞行的关键技术之一。传统的运动规划例如 A*、RRT 等及其改进算法仅为无人机规划出离散的路径点（Waypoints），并没有对无人机的飞行时间、飞行速度、动力学可行性等进行约束，导致无人机依次前往离散的路径点时显得迟钝和机械。此外，预先构建的离线地图不能正常表示场景中的动态物体，这导致无人机在飞行时无法对动态物体进行避让，这将潜在的影响无人机的飞行安全。

本章针对无人机在复杂场景自主飞行所面临的问题，研究了无人机局部运动规划算法，研究思路如图 4-1 所示。首先，本章基于深度相机对局部环境进行栅格地图构建，避免了机载计算机存储大尺度地图造成的内存消耗。其次，基于 B 样条良好的凸包特性，利用 B 样条对轨迹进行参数化，并对无人机飞行轨迹与障碍物之间的距离进行建模。然后，针对无人机在静态中的安全飞行，设计了基于 B 样条的静态场景轨迹生成算法。最后，更进一步考虑了无人机在动态场景的安全飞行，设计了基于简易运动模型的动态场景轨迹生成算法。本章算法能够规划出无人机在静态场景和动态场景中的安全飞行轨迹。

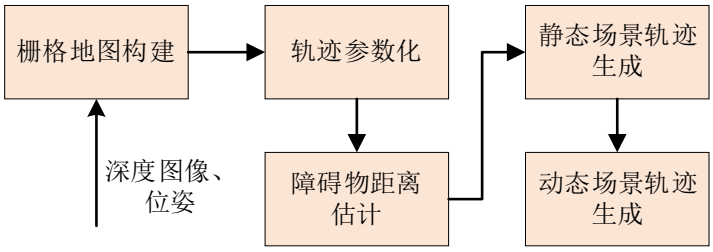


图 4-1 局部运动规划算法研究思路

4.2　局部栅格地图构建

深度相机采集到的点云信息往往难以直接被运动规划系统所使用，且传感器的测量误差也导致对障碍物判断的不准确。因此，一个合适的地图表示方法将有助于降低运动规划的难度以及提升规划的可靠性。栅格地图被广泛运用于机器人运动规划系统中，它以一定的分辨率将连续的物理空间转变为离散的栅格，每个栅格有占据和非占据两种状态，运动规划系统可以直接判断某个栅格是否被占据以确定该栅格是否为障碍物。从最终状态来看，在地图中一个点要么是占据状态（Occupied），要么是自由状态（Free），其状态是确定的。但是，传感器存在误差，单次观测可能发生错误，因此地图中的一个点可能因为观测的不准确导致状态判断错误，因此我们需要多次观测，最终判断一个点的状态。地图中的一个点可以用概率的形式进行表示，通常被障

碍物占据的点的概率表示为$p(s=0)$，自由点的概率表示为$p(s=1)$，两者概率之和为1。引入两者比值作为栅格的状态：

$$Odd(s)=\frac{p(s=1)}{p(s=0)} \tag{3-27}$$

当深度相机观测到新的数据z时，需要对栅格状态进行更新：

$$Odd(s|z)=\frac{p(s=1|z)}{p(s=0|z)} \tag{3-28}$$

表示z发生时，栅格当前的状态。

根据贝叶斯定理，有：

$$p(s=1|z)=\frac{p(z|s=1)p(s=1)}{p(z)} \tag{3-29}$$

$$p(s=0|z)=\frac{p(z|s=0)p(s=0)}{p(z)} \tag{3-30}$$

将式(3-29)与式(3-30)代入式(3-28)，可得：

$$\begin{aligned} Odd(s|z)&=\frac{p(s=1|z)}{p(s=0|z)}\\ &=\frac{p(z|s=1)p(s=1)}{p(z|s=0)p(s=0)}\\ &=\frac{p(z|s=1)}{p(z|s=0)}*\frac{p(s=1)}{p(s=0)}\\ &=\frac{p(z|s=1)}{p(z|s=0)}*Odd(s) \end{aligned} \tag{3-31}$$

式(3-31)两边同时取对数，栅格状态以概率对数进行表示，有：

$$\log Odd(s|z)=\log\frac{p(z|s=1)}{p(z|s=0)}+\log Odd(s) \tag{3-32}$$

由式(3-32)可知，当前的栅格状态$\log Odd(s|z)$可以由本次测量值$\log\frac{p(z|s=1)}{p(z|s=0)}$与上一次的测量值$\log Odd(s)$之和进行表示。

值得注意的是，一个初始状态下的栅格其占据状态概率与自由状态的概率为 0.5，其概率对数为：

$$\log Odd(s_{init})=\log\frac{p(s=1)}{p(s=0)}=\log\frac{0.5}{0.5}=0 \tag{3-33}$$

算法 3 给出了局部栅格地图的更新方式，算法输入为观测值z、判定为占据状态的阈值l_{max}、判定为自由状态的阈值l_{min}、上一次更新的概率对数$\log Odd_{t-1}(s)$。输出为更新后的概率对数$\log Odd_t(s|z)$。

算法 3 栅格地图更新算法

输入： 观测值z, 占据阈值l_{max}, 非占据阈值l_{min}, 前一次观测概率对数$\log Odd_{t-1}(s)$

输出： 更新后的概率对数$\log Odd_t(s \mid z)$

$z \leftarrow \boldsymbol{T}_I^W z$
for each $l_i \in \log Odd_{t-1}(s)$ **do**
  if $l_i \in$z **then**
    $\log Odd_{t,i}(s \mid z) = \log \frac{p(z|s=1)}{p(z|s=0)} + \log Odd_{t-1,i}(s)$
  else
    $\log Odd_{t,i}(s \mid z) = \log Odd_{t-1,i}(s)$
  end if
  if $\log Odd_{t,i}(s \mid z) > l_{max}$ **then**
    $\log Odd_{t,i}(s \mid z) = 1$
  else if $\log Odd_{t,i}(s \mid z) < l_{min}$ **then**
    $\log Odd_{t,i}(s \mid z) = 0$
  end if
end for

在本文中，为了平衡计算复杂度和无人机的感知范围，局部建图范围设定为$5.5m \times 5.5m \times 4.5m$，该算法首先将观测得到的深度点云投影到世界坐标，然后使用一个循环遍历所有栅格，若地图中的一个栅格在新的一次观测中，则利用式(3-32)更新该栅格状态，否则沿用上一次的栅格状态。每次更新后，根据阈值判定该栅格是否为占据状态。栅格地图还需要以无人机半径大小做进一步膨胀处理，好处是无人机在构型空间[84]进行运动规划时可以被视为一个质点，简化了运动规划难度，图 4-2 给出了膨胀处理之后的室内局部栅格地图示意图。

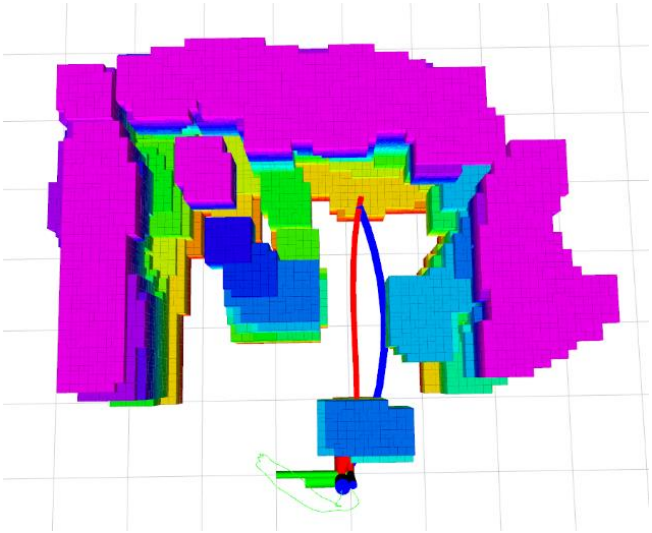


图 4-2 构建的局部栅格地图

4.3 无人机飞行轨迹参数化与障碍物距离估计

4.3.1 无人机飞行轨迹参数化

B 样条常被用于进行曲线设计和曲线拟合，2017 年 Vladyslav[85]将 B 样条用于多

旋翼无人机的运动规划并取得了不错的效果。B 样条曲线由三大要素决定，分别是控制点、节点和阶次，其定义如下：

$$P(t)=\sum_{i=0}^{N}\boldsymbol{P}_i N_{i,k}(t) \tag{3-34}$$

其中，$\boldsymbol{P}_i, i=0,1,\cdots,N$为控制点，$N_{i,k}(t), i=0,1,\cdots,N$为第$i$个$k$阶 B 样条基函数。

B 样条基函数是分段$k+1$阶多项式，由节点向量唯一确定，节点向量$U=\{u_0,u_1,u_2,\cdots,u_m | u_0\leqslant u_1\leqslant u_2\leqslant\cdots\leqslant u_m,\ m=N+k-1\}$为非递减数集合，若节点为均匀等间距分布，则称之为均匀 B 样条。Cox-de Boor 递归公式定义的第i个k阶 B 样条基函数为：

$$\begin{aligned} N_{i,0}(u)&=\begin{cases}1 & \text{if } u_i\leq u<u_{i+1}\\ 0 & \text{else}\end{cases}\\ N_{i,k}(u)&=\frac{u-u_i}{u_{i+k}-u_i}N_{i,k-1}(u)+\frac{u_{i+k+1}-u}{u_{i+k+1}-u_{i+1}}N_{i+1,k-1}(u)\end{aligned} \tag{3-35}$$

B 样条曲线的凸包性质表明，样条曲线包含在控制折线的凸包内。更特别地，如果u在节点区间$[u_i,u_{i+1})$里，那么样条曲线在控制点$\{\boldsymbol{P}_{i-k},\boldsymbol{P}_{i-k+1},\cdots,\boldsymbol{P}_i\}$的凸包里。如图 4-3 所示，飞行轨迹由控制点决定，优化算法使得控制点远离障碍物，飞行轨迹被限制在控制点组成的凸包内，保证了飞行轨迹也在远离障碍物的安全区域。

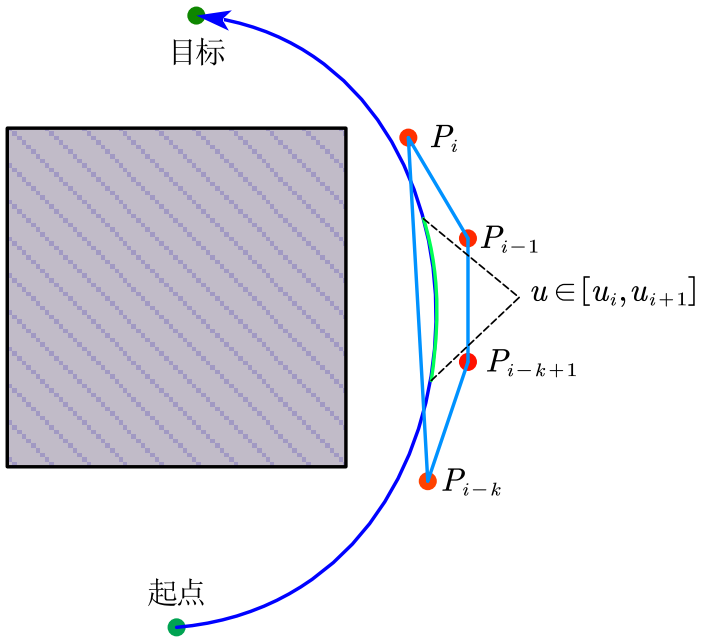


图 4-3 B 样条凸包特性

k阶 B 样条曲线的l阶导数为$k-l$阶 B 样条曲线，这一优良的性质保证了使用 B 样条表示的轨迹的速度、加速度的连续性。B 样条的良好特性使之非常适合无人机的轨迹规划。

为了简化轨迹优化的难度和提升效率，本章将无人机的飞行轨迹参数化为均匀 B 样条，优化变量为控制点$\boldsymbol{P}_i$。B 样条节点为均匀等时间间隔$\triangle t = u_{i+1} - u_i$。那么，飞行轨迹的速度、加速度和急动度可以通过控制点$\boldsymbol{P}_i$差分获得：

$$\boldsymbol{V}_i = \frac{\boldsymbol{P}_{i+1} - \boldsymbol{P}_i}{\triangle t},\ \boldsymbol{A}_i = \frac{\boldsymbol{V}_{i+1} - \boldsymbol{V}_i}{\triangle t},\ \boldsymbol{J}_i = \frac{\boldsymbol{A}_{i+1} - \boldsymbol{A}_i}{\triangle t} \tag{3-36}$$

4.3.2 障碍物距离估计

本章算法为了降低机载计算机的内存占用，并非建立大尺度 ESDF 地图[66]，而是构建局部的栅格地图。但局部栅格地图缺少了 ESDF 地图提供的方向和梯度信息，本节引入一种估计障碍物与飞行轨迹之间距离的方法弥补栅格地图这一缺陷，以指导轨迹优化。

本章算法无需预先规划出初始的无碰撞路径，但需要快速获取障碍物表面信息。根据 A*算法[39]的搜索特点，A*算法输出的可行路径总是紧贴障碍物，如图 4-4 所示。因此利用 A*算法生成的路径近似障碍物表面可降低算法复杂度。

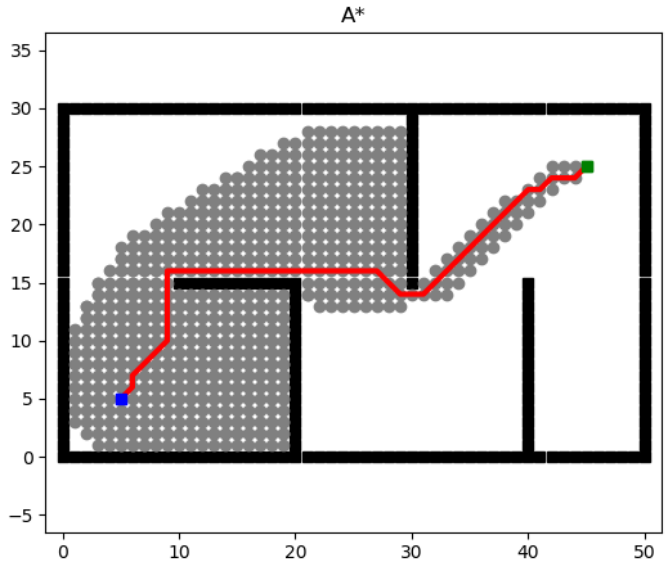


图 4-4 A*算法路径示意图

图 4-5 展示了障碍物距离估计的示意图，左图为无人机飞行轨迹穿越障碍物的俯视图，右图为其中一个控制点$\boldsymbol{P}_i$处的横截面。定义路径Γ为 A*算法得到的无碰撞路径，其紧贴障碍物表面，定义与 B 样条曲线Φ在$\boldsymbol{P}_i$点相切的向量$\boldsymbol{R}_i$为：

$$\boldsymbol{R}_i = \frac{\boldsymbol{P}_{i+1} - \boldsymbol{P}_{i-1}}{2\triangle t} \tag{3-37}$$

然后做垂直于$\boldsymbol{R}_i$的平面Ψ，平面Ψ与无碰撞路径Γ相交$\boldsymbol{p}_i$，$\boldsymbol{v}_i$为$\boldsymbol{P}_i\boldsymbol{p}_i$方向的单位向量，值得注意的是，$\boldsymbol{v}_i$在初次生成之后将不会再改变。那么，控制点$\boldsymbol{P}_i$处的障碍物距离可以表示为：

$$d_i = (\boldsymbol{P}_i - \boldsymbol{p}_i) \cdot \boldsymbol{v}_i \tag{3-38}$$

由式(3-38)可知，当控制点$\boldsymbol{P}_i$在障碍物内部时，距离为负值，当控制点$\boldsymbol{P}_i$在障碍物外部时，距离为正值。障碍物距离的大小以及单位向量$\boldsymbol{v}_i$的方向即可指导优化程序，将控制点“推出”障碍物外，得到安全的飞行轨迹$\boldsymbol{\Phi}'$。

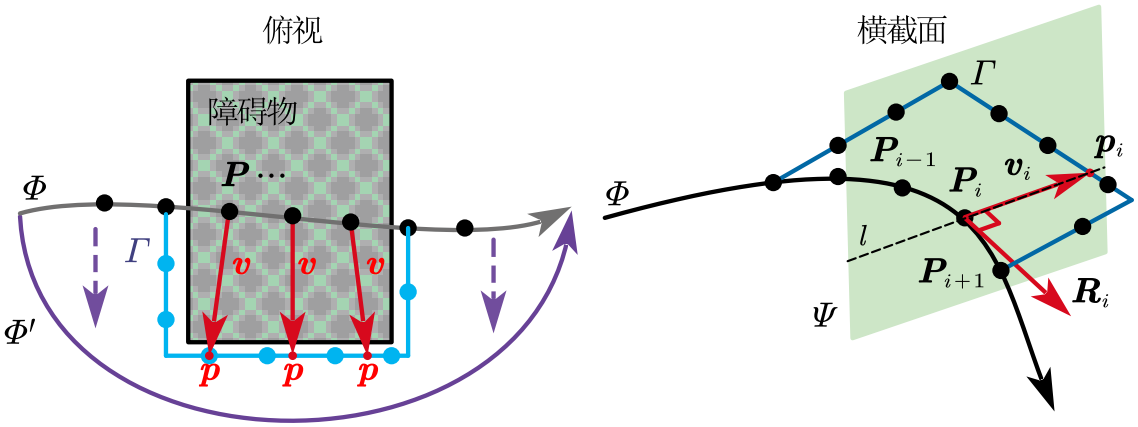


图 4-5 障碍物与轨迹控制点之间的距离估计

4.4　基于样条曲线的安全飞行轨迹生成

根据 2.4.2 节描述的微分平坦模型，为了简化轨迹规划难度，本章算法在四旋翼无人机的微分平坦空间$\sigma=[x,y,z,\psi]^T$中进行规划，即只需规划无人机的位置与偏航角。而实际中的偏航角可通过规划出的速度在水平投影的夹角计算得到，于是只需对无人机位置进行规划。

4.4.1　静态场景安全飞行轨迹生成

为了使飞行轨迹满足无人机的正常安全飞行，定义了包含碰撞代价、轨迹平滑代价、可行性代价在内的罚函数对飞行轨迹进行优化。

（1） 碰撞罚函数

为了使飞行轨迹远离障碍物，定义了安全间隙d_c，表示允许飞行轨迹控制点离障碍物的最近距离。当控制点$\boldsymbol{P}_i$处的障碍物距离d_i满足$d_i>d_c$，则认为由控制点生成的飞行轨迹是安全的。定义碰撞罚函数如式(3-39)所示，该罚函数为分段函数且二阶微分连续，在优化时，罚函数斜率不会随d_i减小而减小，加快了收敛速度。

$$E_c(i)=\begin{cases}0 & (c_i\le 0)\\ c_i^3 & (0<c_i\le d_c)\\ 3d_c c_i^2-3{d_c}^2 c_i+{d_c}^3 & (c_i>d_c)\end{cases} \tag{3-39}$$

其中，$c_i=d_c-d_i$。当$d_i\ge d_c$即$c_i\le 0$，$E_c=0$；当控制点在障碍物表面与安全间隙之间，即$0<c_i\le d_c$，则稍微增大罚函数斜率；当控制点在障碍物内部，即$c_i>d_c$，极大程度增大罚函数斜率迫使轨迹远离障碍物，碰撞罚函数可视化如图 4-6 所示。

所有控制点的碰撞罚函数累加可以得到无人机飞行轨迹整体碰撞罚函数：

$$E_C = \sum_{i=0}^{N} E_c(i) \tag{3-40}$$

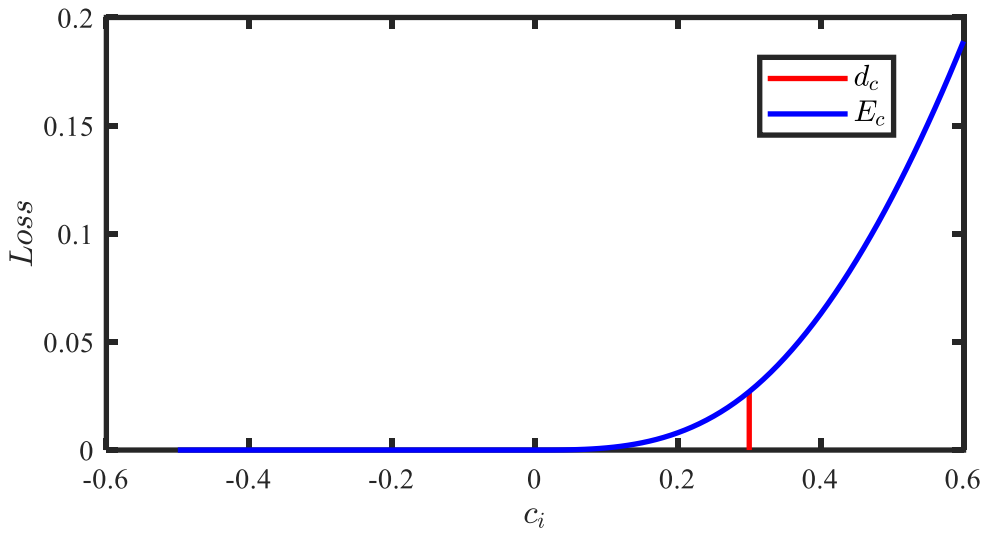


图 4-6 碰撞罚函数可视化

（2） 轨迹平滑罚函数

为了使飞行轨迹尽可能平滑，借鉴了文献[85]中的做法，通过最小化轨迹的 2 阶、3 阶导数的平方和来实现，轨迹平滑罚函数定义如下：

$$E_S = \sum_{i=0}^{N-1} \| \boldsymbol{A}_i \|_2^2 + \sum_{i=0}^{N-2} \| \boldsymbol{J}_i \|_2^2 \tag{3-41}$$

（3） 可行性罚函数

为了确保生成的轨迹可行，必须保证轨迹的速度、加速度以及加加速度有界，构造以下罚函数以确保轨迹的可行性：

$$E_f(\varepsilon) = \begin{cases} 3(\varepsilon^2 - {\lambda_m}^2)^3 & (\varepsilon < -\lambda_m \text{ 或 } \varepsilon > \lambda_m) \\ 0 & (-\lambda_m \leqslant \varepsilon \leqslant \lambda_m) \end{cases} \tag{3-42}$$

其中，$\varepsilon = \{\boldsymbol{V}, \boldsymbol{A}, \boldsymbol{J}\}$，$\lambda_m$ 为各分量限制的最大值。可行性罚函数可视化如图 4-7 所示，当轨迹的有限阶导数不超过限制的最大值时（图中为 0.5），罚函数为 0；当有限阶导数超过限制的最大值时，罚函数导数增大，使得飞行轨迹各阶导数限制在最大值范围之内。

可行性罚函数整体为轨迹各阶导数罚函数加权求和，定义如下：

$$E_F(\varepsilon) = \sum_{\varepsilon = \{\boldsymbol{V}, \boldsymbol{A}, \boldsymbol{J}\}} \lambda_\varepsilon E_f(\varepsilon) \tag{3-43}$$

其中，λ_ε 为轨迹各阶导数罚函数的权重。

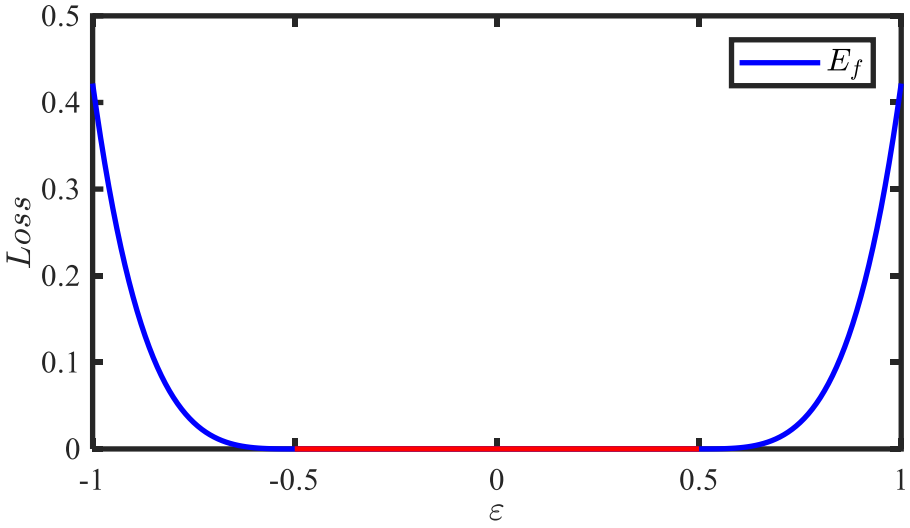


图 4-7 可行性罚函数可视化

至此，构建无人机在静态场景下安全飞行轨迹的整体罚函数如(3-44)所示：

$$E_{static} = w_S E_S + w_F E_F + w_C E_C \tag{3-44}$$

其中，w_S, w_F, w_C 为各项罚函数权重。

4.4.2　引入简易运动模型的动态障碍物规避

无人机自主飞行时，视野中出现动态障碍物时往往难以主动规避，主要体现在 SLAM 模块输出误差较大的位姿信息导致无人机轨迹跟踪不准确，以及运动规划模块并没有对动态障碍物运动进行建模。其中 SLAM 模块在动态场景中的位姿估计问题已经在第 3 章给出了可行的解决方案，使得 SLAM 模块输出的障碍物信息和位姿在本章可以直接使用。

考虑到无人机自主飞行的前向速度较快，本章仅将动态障碍物建模为平行于相机平面的运动，如图 4-8 所示。

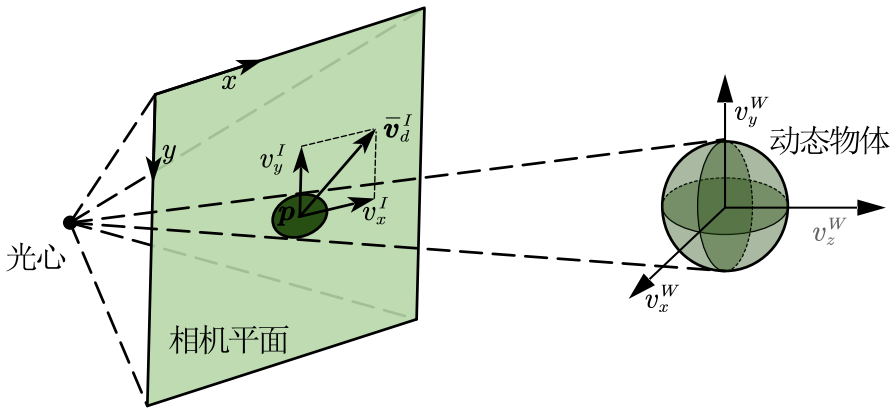


图 4-8 动态障碍物运动建模示意图

在无人机视场范围内的动态障碍物可由 SLAM 模块的超轻量目标检测算法 PicoDet 检出，并通过前后两帧计算出像素坐标系下的动态障碍物速度：

$$\bar{\boldsymbol{v}}_d^I = \frac{\boldsymbol{p}_{i+1} - \boldsymbol{p}_i}{1/fps} \tag{3-45}$$

其中，$\boldsymbol{p}_i$为 PicoDet 输出的第i帧图像中动态障碍物检测框中心坐标。

并将速度变换到世界坐标系下：

$$\bar{\boldsymbol{v}}_d^W = \boldsymbol{T}_{(1:3)}{}^{-1}\boldsymbol{K}^{-1}\bar{\boldsymbol{v}}_d^I \tag{3-46}$$

其中，$\boldsymbol{T}_{(1:3)}$为世界坐标系到相机坐标系的坐标转换的非齐次表示，$\boldsymbol{K}$为相机内参。那么，动态障碍物简化后的运动轨迹可以表示为式(3-47)。

$$\boldsymbol{\Phi}_d(t) = \bar{\boldsymbol{v}}_d^W t \tag{3-47}$$

图 4-9 给出了无人机进行动态障碍物规避的示意图，从无人机视场出现动态障碍物的时刻t_s至动态障碍物离开无人机视场的时刻t_e，无人机轨迹$\boldsymbol{\Phi}_f(t)$与动态障碍物轨迹$\boldsymbol{\Phi}_d(t)$之间的距离可以被定义为：

$$d(t) = \|\boldsymbol{\Phi}_f(t) - \boldsymbol{\Phi}_d(t)\| \tag{3-48}$$

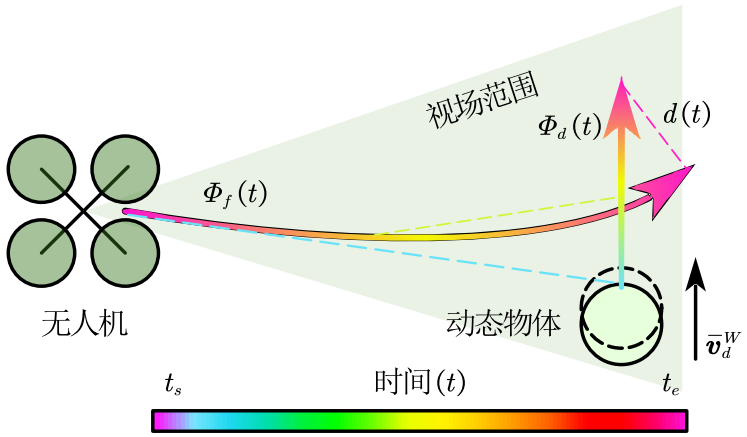


图 4-9 动态障碍物规避示意图

参考文献[53]中的做法，构造无人机与动态障碍物之间罚函数如(3-49)所示，使无人机飞行轨迹规避动态障碍物。

$$E_{dynamic} = \begin{cases} \int_{t=t_s}^{t_e} (d(t) - C)^2 dt & d(t) < C \\ 0 & d(t) \geq C \end{cases} \tag{3-49}$$

其中，C为无人机与动态障碍物之间的安全距离，动态设置为检测框宽度的 1.2 倍。

至此，结合静态场景罚函数E_{static}与动态场景罚函数$E_{dynamic}$，并使用 L-BFGS 算法[86]对轨迹进行优化，如式(3-50)所示，即可得到无人机在复杂动态场景中的安全飞行轨迹。

$$\hat{\boldsymbol{P}} = \underset{\boldsymbol{P}}{arg\min}\,(E_{static} + E_{dynamic}) \tag{3-50}$$

4.5　实验与分析

本章研究的运动规划算法为局部运动规划算法，无人机无法获得全局地图信息，当障碍物的大小超过本章设定的局部建图范围时，局部运动规划算法将不能生成绕过障碍物的安全飞行轨迹，即运动规划失败。除此之外，本章研究的局部运动规划算法无需障碍物满足特定约束，本小节将会在静态场景和动态场景中随机生成多种不超过局部建图范围的障碍物，以及在不同密度的障碍物场景下进行实验。

考虑到计算复杂度，设置局部建图与规划范围为$5.5m \times 5.5m \times 4.5m$，设置本章算法的 B 样条阶次$k=3$，控制点个数$N=25$。为了验证本章运动规划算法的性能，在多种场景中进行了仿真实验并与 fast-planner[87]进行比较。用于进行仿真实验的电脑中央处理器为 AMD Ryzen7 4800H，主频 4.2GHz，显卡为 Nvidia RTX2060（仅用于仿真时渲染画面，不参与算法计算），操作系统为 Ubuntu18.04，搭载 ROS 机器人操作系统，可视化软件使用 ROS 组件 Rviz。仿真实验所用的地图采用开源软件 Mockamap[88]进行生成。

4.5.1　静态场景仿真实验

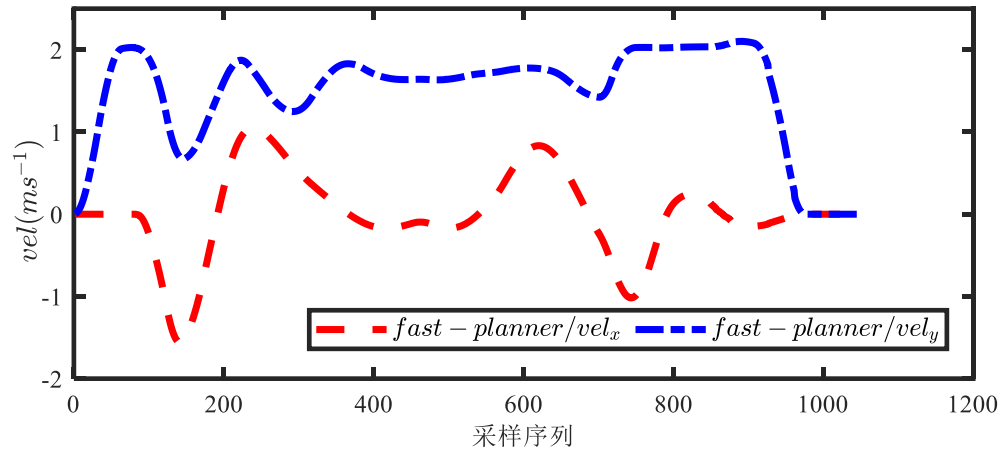
（1） 2D 静态场景仿真实验

为了验证本算法在静态场景中的有效性，在采用 Mockamap 生成密度为 0.18 个$/m^2$静态障碍物的随机森林中进行 2D 运动规划，设定无人机最大飞行速度为$2m/s$，最大加速度为$6m/s^2$，起点为$[0m, -16m]^T$，终点为$[0m, 0m]^T$，飞行轨迹如图 4-10 所示。

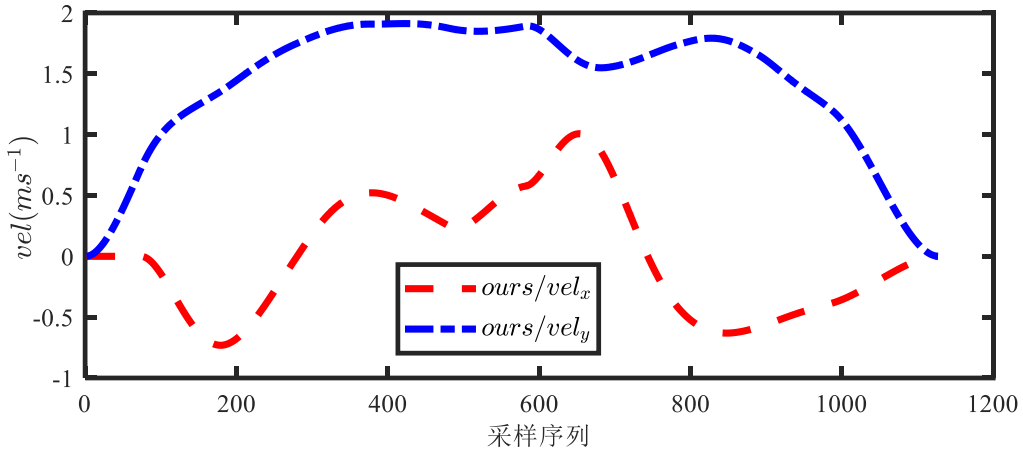
图 4-10 2D 静态场景轨迹对比图

图 4-10(a)中，绿色圆圈标记出了 fast-planner 生成的飞行轨迹距离障碍物较近。由于本章算法利用平滑度罚函数对轨迹平滑度进行优化，因此生成的轨迹也更为平滑。

图 4-11 给出了两种算法生成轨迹的速度对比，其中横坐标表示采样序列，并不代表实际采样时间。飞行速度方面，本章算法的飞行速度更为稳定，在y轴方向（前进方向），飞行速度大部分时间接近设定的最大值，平均飞行速度为$1.42m/s$，而 fast-planner 的飞行速度波动较大，平均飞行速度为$1.34m/s$。飞行时间方面，从起点到终点本文算法飞行时间为$11.2s$，fast-planner 则为$11.9s$。



(a) fast-planner 速度曲线图



(b) 本章算法速度曲线图

图 4-11 2D 静态场景飞行速度对比

（2） 3D 静态场景仿真实验

为了更进一步展示本章算法的有效性，在采用 Mockamap 生成密度为0.16个$/m^2$、最大高度$2m$的 3D 静态障碍物场景进行仿真实验，设定两种算法最大速度$2m/s$，最大加速度$3m/s^2$，起点为$[3m, -13m, 1m]^T$，终点为$[0m, 2m, 0m]^T$，飞行轨迹如图 4-12 所示。图中可以看到，两种算法在遇到比初始位置更高的障碍物时都能调整飞行

高度避开障碍物，在该场景下都能生成安全的飞行轨迹，但 fast-planner 高度变化更为剧烈。

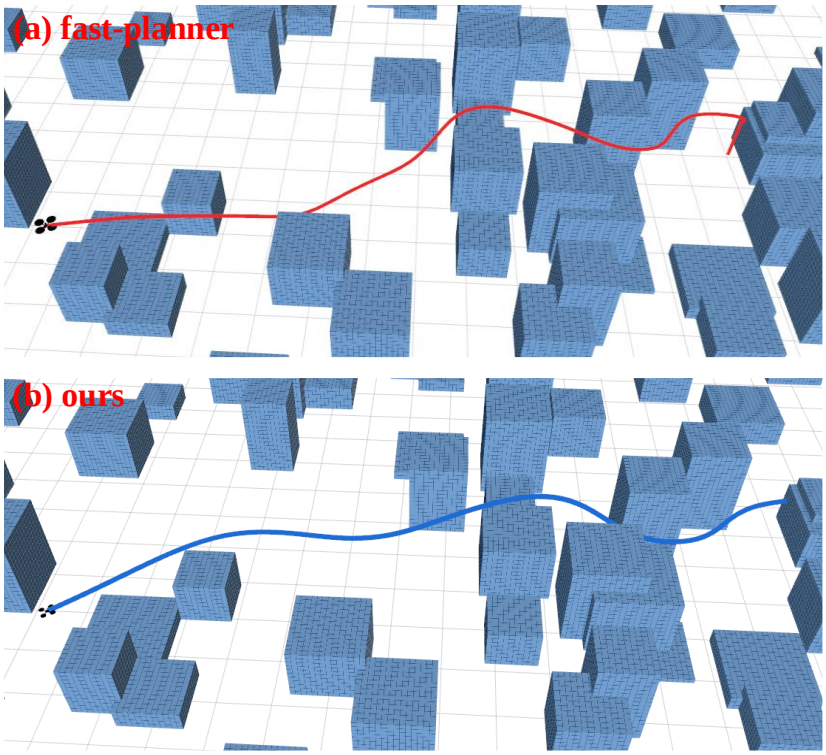
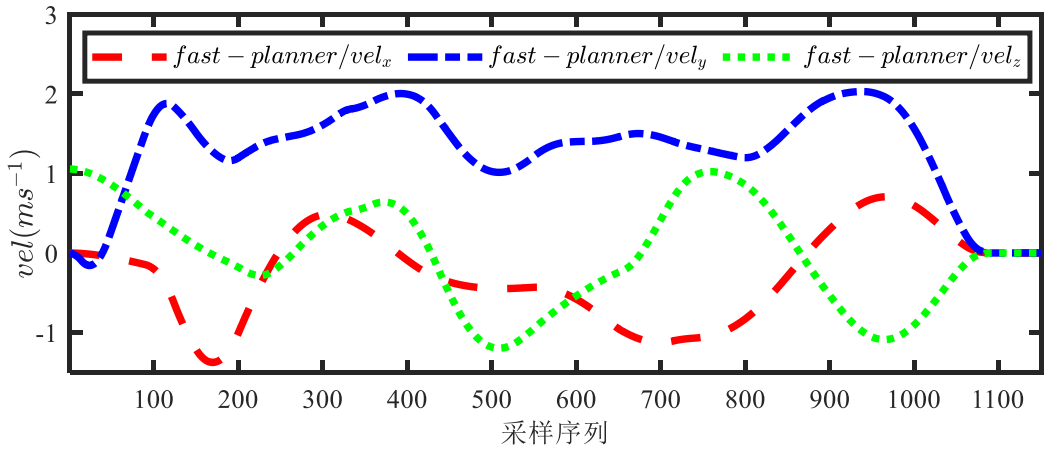
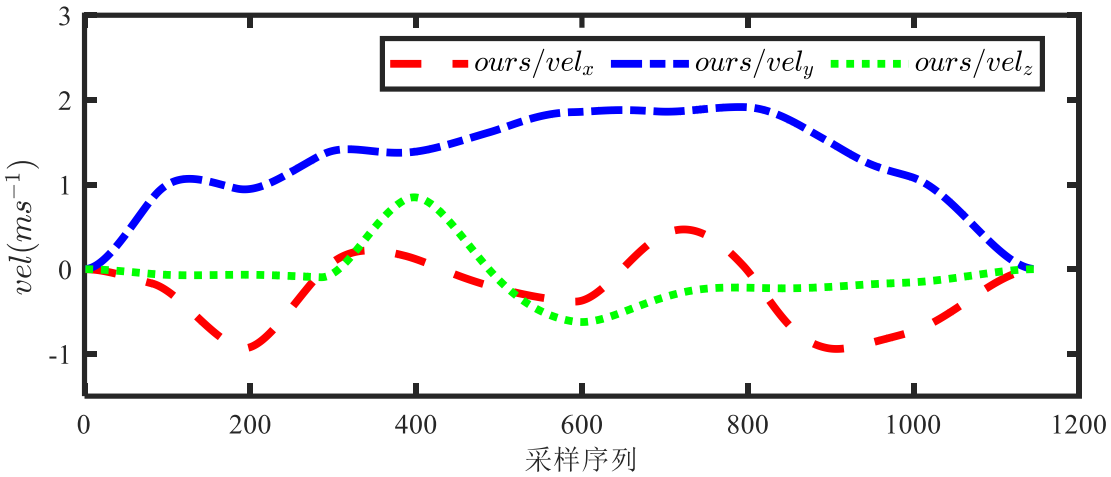


图 4-12 3D 静态场景飞行轨迹对比

图 4-13 给出了两种算法生成轨迹三轴速度对比。本章算法的y轴（前进方向）速度更为平稳，在较长时间接近最大飞行速度$2m/s$，平均速度为$1.36m/s$，在飞行$10.9s$后到达终点；而 fast-planner 在多次达到最大飞行速度后又迅速下降，平局速度为$1.30m/s$，飞行时间为$11.4s$，说明本章算法在前方遇到障碍物时能够更快做出反应，使得y轴方向速度相对更加稳定。本章算法的x轴与z轴速度变化也更加平缓，说明本章算法的飞行轨迹更加平滑。



(a) fast-planner 速度曲线图

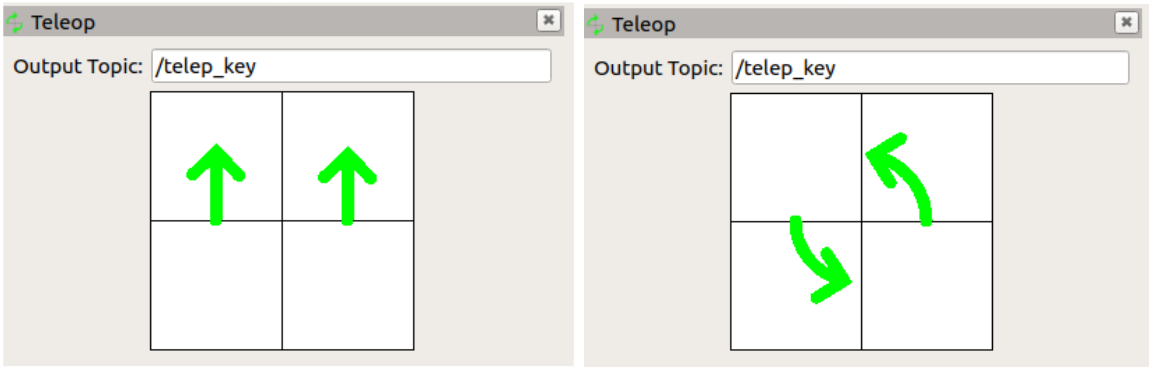


(b) 本章算法速度曲线图

图 4-13 3D 静态场景飞行速度对比

4.5.2 动态场景仿真实验

为了验证本章算法在含有动态障碍物场景下的飞行性能，在无人机飞行途中分别放置单个动态障碍物和两个动态障碍物并进行飞行实验。由于 fast-planner 需要 ESDF 地图更新，不适用于动态场景[89]，因此动态场景实验只针对本章实现的算法进行验证。由于在仿真环境中无法获得 RGB 数据，无法通过第三章算法获得潜在动态对象的检测框，因此无法利用本章算法得知动态障碍物的移动速度，在本次动态场景仿真实验中，则直接将障碍物的移动速度传递给本章运动规划算法，无人机对动态障碍物的规避策略仍然采用本文在 4.4.2 小节给出的方法。在此实验中，无人机起点设置为 $[0m, -13m, 1m]^T$，终点设置为 $[0m, 0m, 1m]^T$，最大速度为 $2m/s$，最大加速度 $6m/s^2$，动态障碍物的移动速度和方向通过类型为“geometry_msgs/Twist”的话题“/telep_key”进行控制，如图 4-14 所示。



(a) 直线移动　　(b) 转弯移动

图 4-14 仿真中障碍物移动控制

（1） 单动态障碍物场景实验

图 4-15 给出了本章算法在单个动态障碍物场景中进行实验的其中三次飞行轨迹。

可以看到，无人机飞行轨迹不仅能够安全穿越静态障碍物，而且在接近动态障碍物时，轨迹存在明显避让。图 4-15(a)中，无人机在对动态障碍物进行避让的飞行途中存在静态障碍物，本章算法能够生成安全的飞行轨迹。

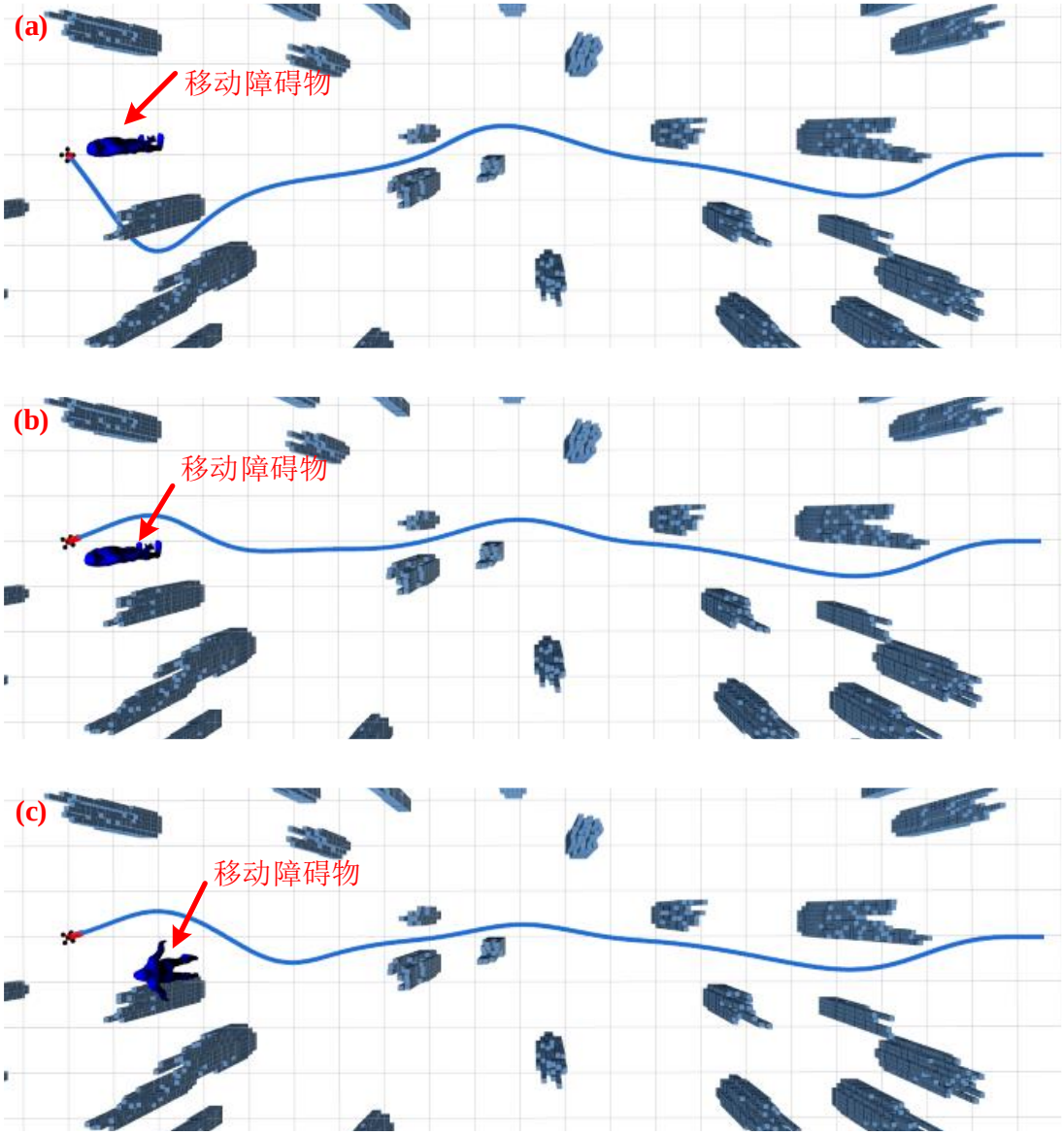


图 4-15 本章算法在单个动态障碍物场景中的飞行轨迹

（2） 两个动态障碍物场景实验

图 4-16 给出了本章算法在两个动态障碍物场景中进行实验的其中三次飞行轨迹。可以看到，本章算法对飞行途中接连出现两个动态障碍物的场景仍然适用。从三次飞行轨迹可以看到，无人机在动态场景中飞行轨迹具有良好的平滑度，根据动态障碍物移动状态，生成了较为平滑的飞行轨迹。

从动态场景的仿真实验可以发现，本章设计的基于简易运动模型的动态障碍物轨迹算法无论是在单个动态障碍物的场景或者两个动态障碍物的场景，都能与动态障碍保持安全间距，规划出平滑安全的飞行轨迹。

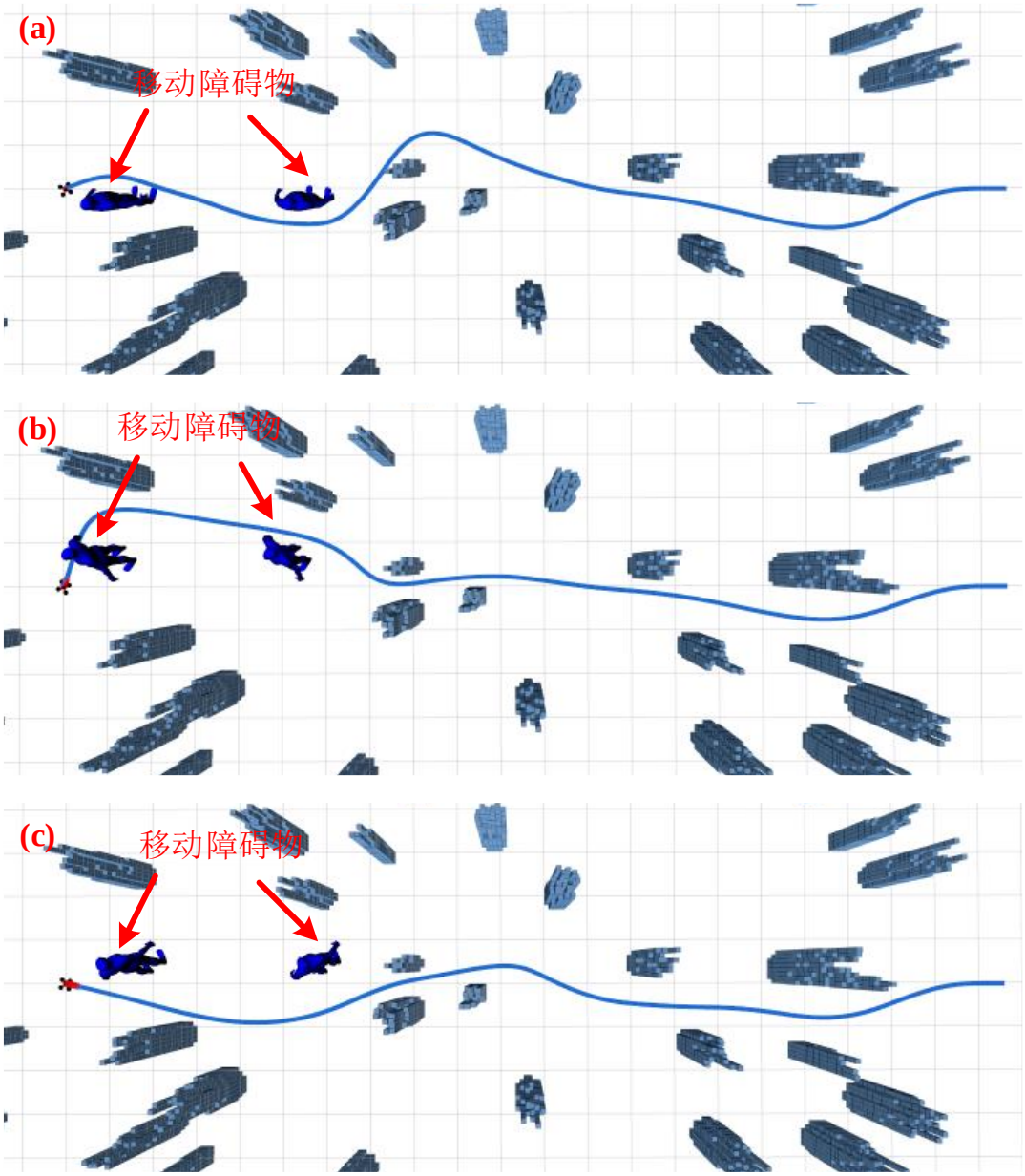


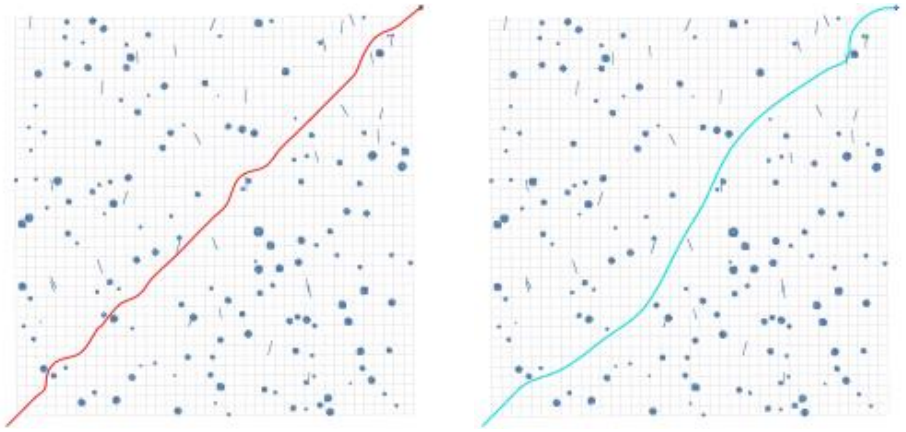
图 4-16 本章算法在两个动态障碍物场景中的飞行轨迹

4.5.3 不同障碍物密度实验

为了更进一步验证本章算法在不同障碍物密度环境下的有效性，在不同障碍物密度环境中进行仿真实验。使用开源随机地图生成器生成大小为$40\times40\times3m$、密度分别为 0.1、0.15、0.20、0.25 个$/m^2$障碍物的随机地图。障碍物高度范围为 0~3m，障碍物之间最小距离为 $0.8m$。无人机运动规划的起点设定为$[-21m,-21m,1m]^T$，终点设定为$[21m,21m,1m]^T$，最大飞行速度设定为$5m/s$，最大加速度为$3m/s^2$。

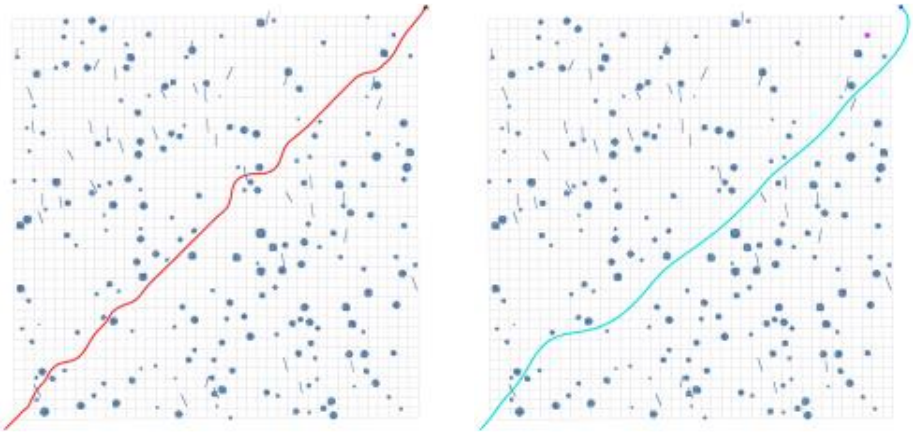
在不同密度障碍物场景中，分别对 fast-planner 和本章算法进行各 5 次仿真实验，每种密度障碍物场景下的其中一次飞行轨迹俯视图如图 4-17 至图 4-20 所示。图中蓝色部分为障碍物，红色线条为 fast-planner 仿真得到的飞行轨迹，青色线条为本章算法得

到的飞行轨迹。对不同密度障碍物场景中的 5 次仿真飞行得到的评估指标求均值，结果见表 4-1 所示。



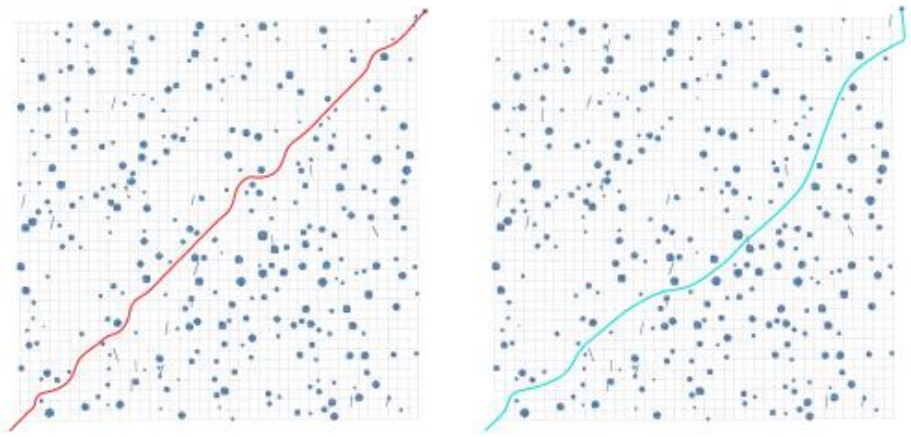
(a) Fast-planner　　(b) 本章算法

图 4-17 密度为 0.1 的障碍物环境下飞行轨迹对比



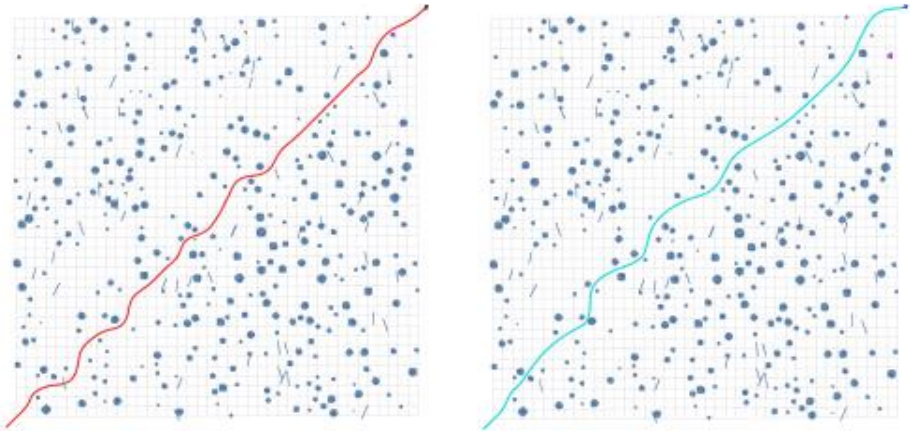
(a) Fast-planner　　(b) 本章算法

图 4-18 密度为 0.15 的障碍物环境下飞行轨迹对比



(a) Fast-planner　　(b) 本章算法

图 4-19 密度为 0.20 的障碍物环境下飞行轨迹对比



(a) Fast-planner　　(b) 本章算法

图 4-20 密度为 0.25 的障碍物环境下飞行轨迹对比

表 4-1 不同障碍物密度实验平均指标对比

算法	障碍物密度（个/m^2）	花费时间（s）	飞行长度（m）	最大速度（m/s）	平均速度（m/s）
fast-planner[87]	0.10	21.1	59.5	3.8	2.8
	0.15	21.0	59.5	3.8	2.8
	0.20	21.3	59.7	3.7	2.8
	0.25	20.6	59.7	4.0	2.9
本章算法	0.10	15.6	65.8	4.6	4.2
	0.15	16.5	68.8	4.4	4.1
	0.20	19.2	67.4	4.5	3.5
	0.25	20.2	67.4	4.6	3.3

从图 4-17 至图 4-20 以及表 4-1 可知，fast-planner 与本章算法都能够在 0.1、0.15、0.20、0.25个/m^2障碍物的场景中生成安全飞行轨迹，但本章算法能够生成的飞行轨迹更为平滑，体现为本章算法倾向于绕行“大圈”，而 fast-planner 在绕行障碍物后返回之前的路径。本章算法生成的安全飞行轨迹速度较快，在所有场景中的平均飞行速度约为$3.8m/s$，略快于 fast-planner 算法的$2.8m/s$。在 0.25个/m^2障碍物的复杂场景中最大速度可达$4.6m/s$。在时间方面，本章算法的飞行时间随障碍物密度增大而逐渐变长，而 fast-planner 的飞行时间无论是低密度还是高密度的障碍物场景都表现得相对稳定。在飞行路径长度方面，本章算法产生更为平滑的轨迹，绕行障碍物较多，因此飞行路径长度比 fast-planner 更长。

4.6 本章小结

本章对多旋翼无人机自主飞行的关键技术之一——运动规划算法展开研究。针对传统路径规划算法难以在复杂环境下为无人机生成安全飞行轨迹的问题，本章研究了基于样条曲线的多旋翼无人机局部运动规划算法。为了减小地图构建时间和内存消耗，使用深度信息构建障碍物环境的局部栅格地图，并设计了障碍物与轨迹之间距离的表示方法。鉴于 B 样条良好的凸包特性，使用 B 样条对轨迹进行参数化。为了在静态场景中的生成安全的飞行轨迹，构建了碰撞罚函数、平滑性罚函数、可行性罚函数对轨迹进行数值优化。此外，设计了基于简易运动模型的动态障碍物规避方法。在静态场景、动态场景和不同障碍物密度场景进行了仿真实验，结果表明本章算法能够在复杂环境中生成安全的无人机飞行轨迹。

5　自主飞行系统构建与验证

5.1　引言

前文对视觉数据驱动的多旋翼无人机自主飞行任务中的其中两个关键技术开展了研究。本文第三章提出了一种基于静态特征点筛选的动态环境 RGBD SLAM 算法。本文第四章对多旋翼无人机局部运动规划算法展开研究，并对局部运动规划算法进行了仿真实验，实现了多旋翼无人机在多种静态场景和动态场景中的安全轨迹生成。本章将结合视觉 SLAM 的位姿信息以及运动规划算法生成的安全飞行轨迹，构建多旋翼无人机自主飞行系统，并对其进行实验，进一步测试本文算法在复杂动态场景中的定位精度与稳定性。

在本章中，首先对多旋翼无人机硬件进行选型和设计，然后基于 ROS 机器人操作系统设计了自主飞行软件和算法，最后基于构建的自主飞行系统在真实环境中分别进行 SLAM 真实环境实验、无人机定点悬停实验以及自主飞行实验。实验结果表明，该系统能够在真实场景中安全稳定飞行。

5.2　系统设计与实现

5.2.1　系统平台搭建

本文搭建的四旋翼无人机系统结构如图 5-1 所示，包括负责定位与轨迹生成的感知与决策硬件、负责无人机姿态控制、轨迹跟踪控制的运动控制硬件以及监控各类信息的地面站硬件。其中机载计算机 CPU 为 I7-8565U，主频 1.8GHz~4.6GHz，内存 8G，功耗 40W，以提供该系统运行 SLAM 算法和运动规划算法所需算力。

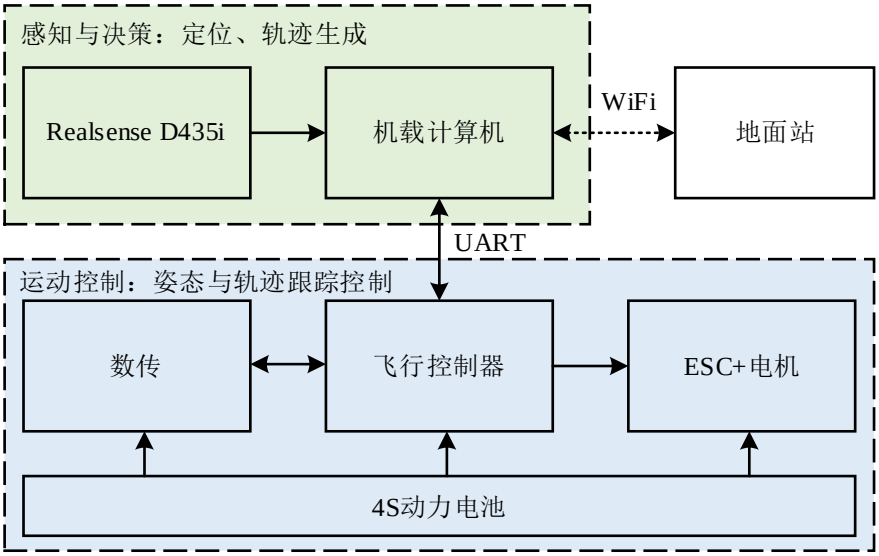


图 5-1 系统结构图

搭建的自主飞行无人机系统实物如图 5-2 所示。整机重量 1200g，最大推力 4400g，两电机之间的距离仅为 18cm，贴在机身表面的 4 颗反光球能够在运动捕捉系统(Motion Capture System)中为整个系统提供真实位姿和轨迹。

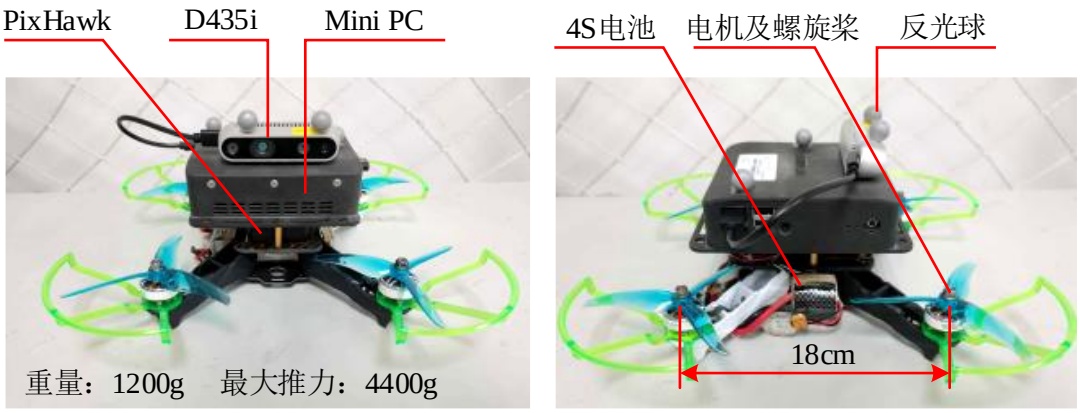


图 5-2 实验平台

该系统中与动力、算力相关的主要硬件如表 5-1 所示。

表 5-1 实验系统主要硬件型号

硬件	型号
飞行控制器	Pixhawk 2.4.6
电机	T-Motor 2306 KV2400
电子调速器	LANRC 45A DSHOT600
螺旋桨	T-Motor 5146
动力电池	格氏 4S 2300mAh
视觉传感器	英特尔 Realsense D435i
机载计算机	I7-8565U

为了验证本系统的性能，采用北京航空航天大学的飞行评测工具[90]进行评测，得到的部分理论性能参数如表 5-2 所示。可以看到，本文搭建的实验系统的可悬停时间充裕，油门余量足，飞行速度快，机动性强，满足本文实验需求。

表 5-2 系统平台性能参数

性能参数	值
悬停时间	6.4min
悬停油门百分比	37.5%
剩余负载	2.46Kg
最大油门总升力	42.4N
最大飞行速度	14.1m/s
单程飞行距离	5.16Km

5.2.2　相机参数标定

（1） 相机内参标定与深度相机校准

在本系统中，Realsense D435i 的帧率设定为$30fps$，分辨率640×480。为了在后续 SLAM 算法的正常运行，需要对 RGB 相机进行标定。本文固定标定板的空间位置不变，在 6 个自由度的多次移动相机拍摄标定板照片共 16 张，其中部分拍摄角度如图 5-3 所示。

图 5-3 相机标定图像采集

然后利用张氏标定法[91]对 D435i 的 RGB 相机进行标定，得到的相机的内部参数如式(4-1)所示：

$$\boldsymbol{K}=\begin{bmatrix}608.188 & 0 & 322.615\\ 0 & 608.520 & 234.001\\ 0 & 0 & 1\end{bmatrix} \tag{4-1}$$

相机径向畸变和切向畸变为：

$$\begin{aligned}[k_1,k_2,k_3]&=[0.0313,0.7509,-2.8533]\\ [p_1,p_2]&=[0.0024,-7.8489e^{-04}]\end{aligned} \tag{4-2}$$

每幅图像重投影平均重投影误差如图 5-4 所示，可见误差均小于 0.2 个像素，满足实验系统需求。

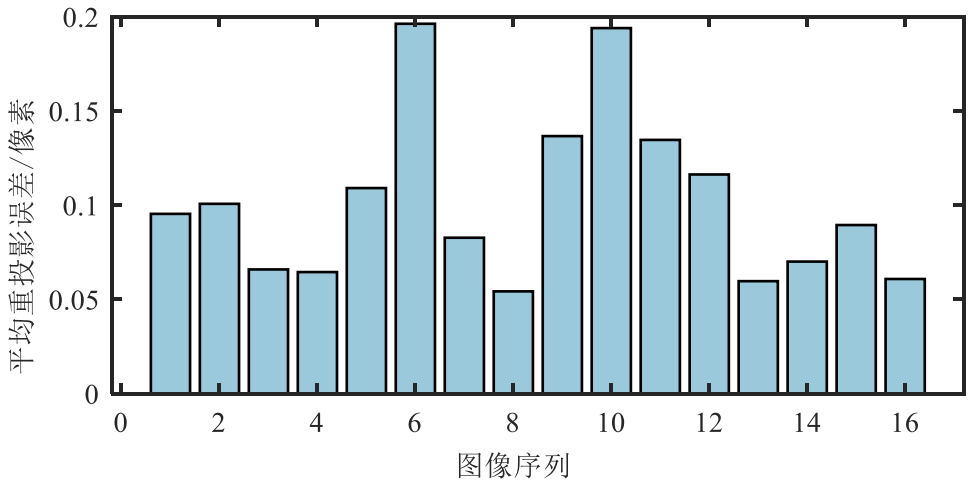


图 5-4 每幅图像的重投影误差

Realsense D435i 的深度相机则采用 Intel 官方开发的 Depth Quality Tool 软件进行校准。

（2） RGBD 相机与无人机质心外参标定

视觉数据驱动的多旋翼无人机自主飞行时需要对无人机进行定位，本文视觉 SLAM 算法利用 RGBD 相机作为数据源，算法输出的位姿为相机的位姿，而非多旋翼无人机的位姿。RGBD 相机安装时与无人机质心存在固定的平移和旋转，当求得 RGBD 相机与无人机质心的变换矩阵时，无人机在空间中的位姿可由 RGBD 相机位姿变换得到。

考虑到无人机飞行控制器安装方式，认为 IMU 位于无人机质心，于是可通过相机与 IMU 外参标定得到两者的位姿关系。设定相机帧率为$30Hz$，飞行控制器 IMU 帧率为$200Hz$，手持多旋翼无人机绕x、y和z轴旋转和移动，对 AprilGrid 标定板[92]和 IMU 数据进行同步采集，如图 5-5 所示。

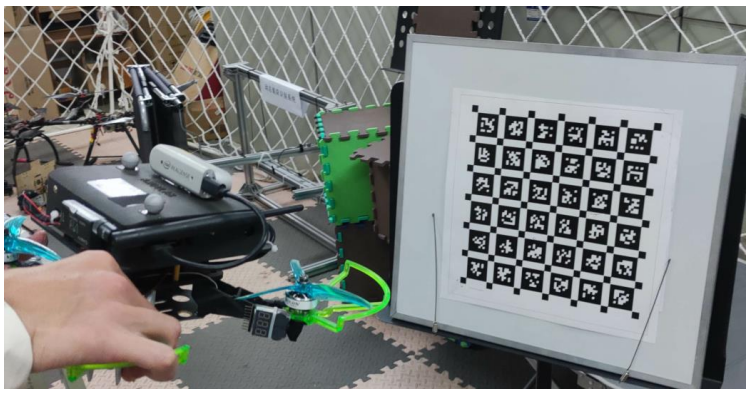
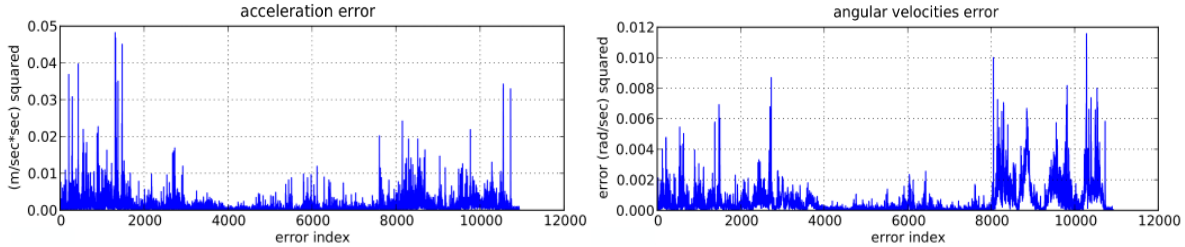


图 5-5 相机-IMU 标定数据采集场景

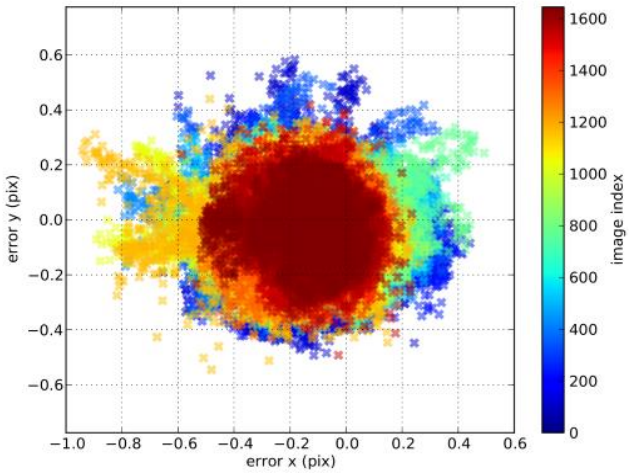
使用 Kalibr 工具箱[93]对 RGB 相机和 IMU 之间的外部参数进行联合标定，标定得到的相机于 IMU 的外参为：

$$\boldsymbol{T}_C^I = \begin{bmatrix} -0.0267 & -0.1987 & 0.9797 & -0.0772 \\ -0.9997 & 0.0141 & -0.0244 & 0.0374 \\ -0.0090 & -0.9800 & -0.1990 & 0.0817 \\ 0 & 0 & 0 & 1 \end{bmatrix} \tag{4-3}$$

相机的平均重投影误差为0.0474，陀螺仪均方根误差为0.0006，加速度计均方根误差为0.0014，满足系统需求，标定误差可视化如图 5-6 所示。



(a) 加速度误差与角速度误差



(b) 图像重投影误差

图 5-6 标定误差可视化

5.2.3 软件设计与实现

考虑到本文实验系统软件模块多、模块间信息交互较为复杂，因此本文实验系统所有软件基于 Ubuntu18.04 下的 ROS 进行实现。利用 ROS 进行机器人软件开发，不同功能可以划分为不同的程序，程序间通过话题（Topic）、服务（Service）等消息传递方式进行通信，这种松耦合的方式降低了机器人软件开发的难度。本文实验系统需要机载计算机运行自主飞行算法、地面计算机进行监控以及 Mocap 系统获取轨迹真值，因此需要在多台计算机进行数据传输，ROS 的跨平台、跨设备的通信机制为本文实验系统提供了有力支持。

图 5-7 给出了本文实验系统软件主体部分的框架图。主要分为机载计算机软件、地面站软件和运动捕获系统软件，如图中虚线框所示。图中椭圆表示 ROS 节点（Node），即完成某一特定功能的程序，方形表示 ROS 话题（Topic），为封装了不同消息或者指令的数据结构，不同节点之间通过订阅与发布话题的方式进行通信。机载计算机运行 RGBD 相机的驱动节点 Realsense-ros、动态环境 SLAM 节点、运动规划节点以及与 PixHawk 飞行控制器通信的节点 MAVROS。动态环境 SLAM 节点订阅 Realsense-ros 发布的原始图像和深度图像话题用以无人机位姿估计；运动规划节点订阅 Realsense-ros 发布的深度图像话题以及动态环境 SLAM 节点输出的里程计信息与动态对象预选框，用以规划出无人机在障碍物环境中的安全飞行轨迹；MAVROS 节点订阅运动规划节点输出的位姿控制指令并将指令通过串口下发至 PixHawk 飞行控制软件以控制无人机跟随运动规划模块规划的轨迹。地面站软件主要作用为监测实验系统各项数据以及向实验系统发送启动、停止指令等。运动捕获系统软件运行在独立工作站上，主要作用为通过多旋翼无人机上粘贴的反光球捕获无人机位姿，为后续评估 SLAM 算法定位精度提供真实位姿。

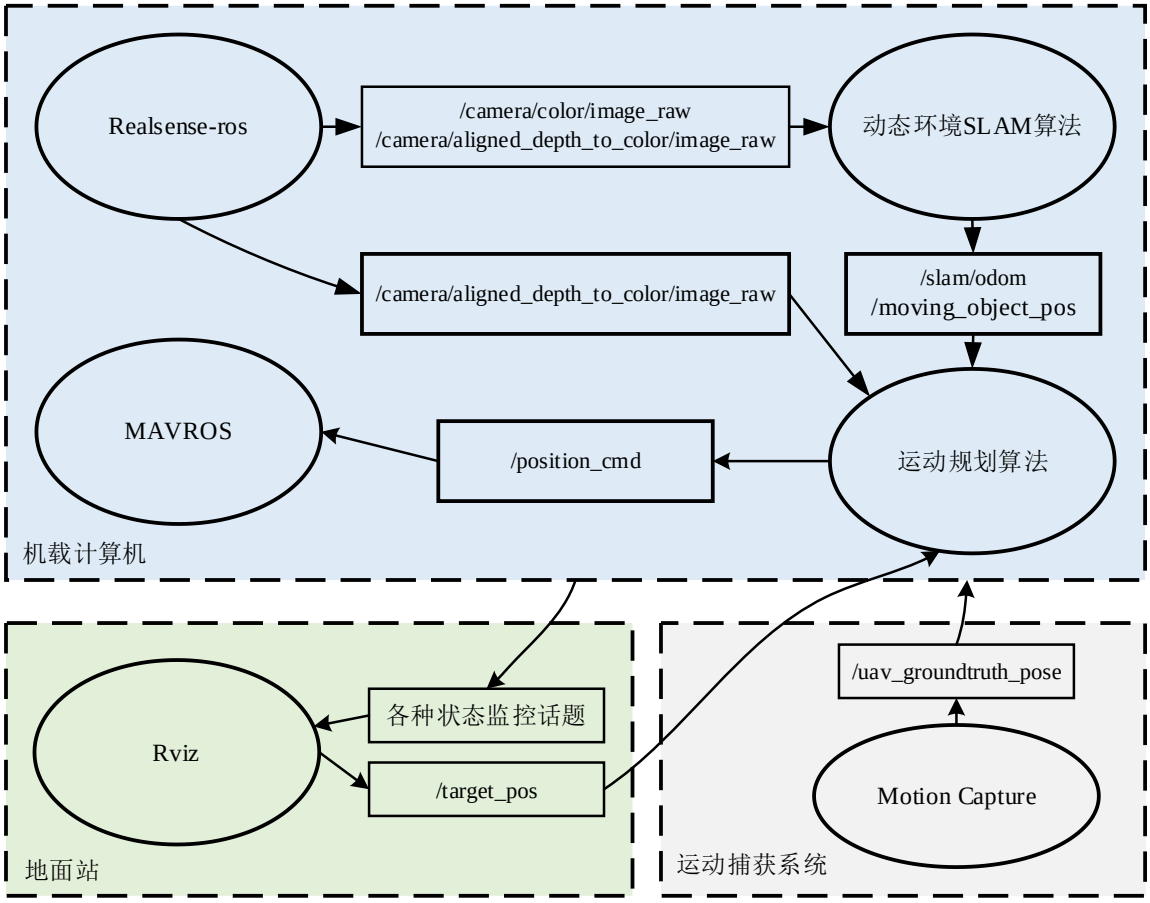


图 5-7 系统软件架构图

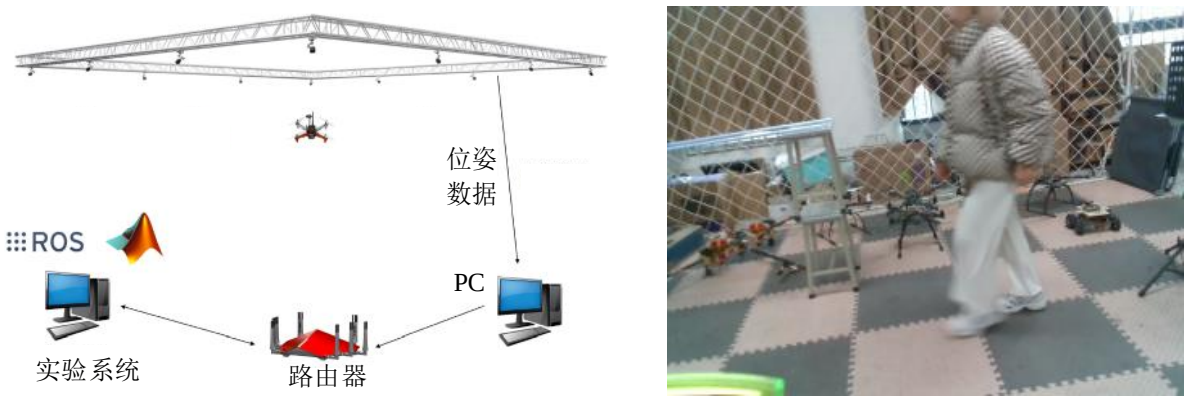
5.3　系统验证

在第三章和第四章，分别对动态环境下的 RGBD SLAM 以及多旋翼局部运动规划算法进行了实验验证，证明了算法在复杂环境中的可行性。本章将以上两章算法相结合，构建完整的视觉数据驱动的多旋翼无人机自主飞行系统，进一步验证本文算法在真实复杂环境中的可行性。首先在真实动态场景中测试本文提出的动态环境下 RGBD SLAM 系统的定位精度。其次测试以动态环境下的 RGBD SLAM 为唯一定位源的自主飞行系统的定点悬停精度。最后在室内外场景验证自主飞行系统的可行性。

5.3.1　SLAM 真实动态场景实验

为了在真实场景中评估本文基于静态特征点筛选的动态环境 RGBD SLAM 算法，手动操控本章构建的多旋翼无人机在室内 OptiTrack 运动捕获系统[94]（Motion Capture System）下的动态场景中进行飞行，同步采集原始 RGB 图像、深度图像和运动捕获系统输出的无人机真实位姿。OptiTrack 运动捕获系统通过在室内安装多个摄像头，同时拍摄运动目标上安装的反光球，通过多视图几何解算出多个 Marker 构成的刚体的位姿，然后再通过局域网发送到实验系统，定位精度可达亚毫米级。OptiTrack 运动捕获系统与数据采集场景如图 5-8 所示，图 5-8(a)为运动捕获系统数据流示意图，图 5-8(b)为无

人机在室内运动捕获系统下飞行采集的动态场景。



(a) 动捕系统数据流　　(b) 数据采集场景

图 5-8 OptiTrack 运动捕获系统与数据采集场景

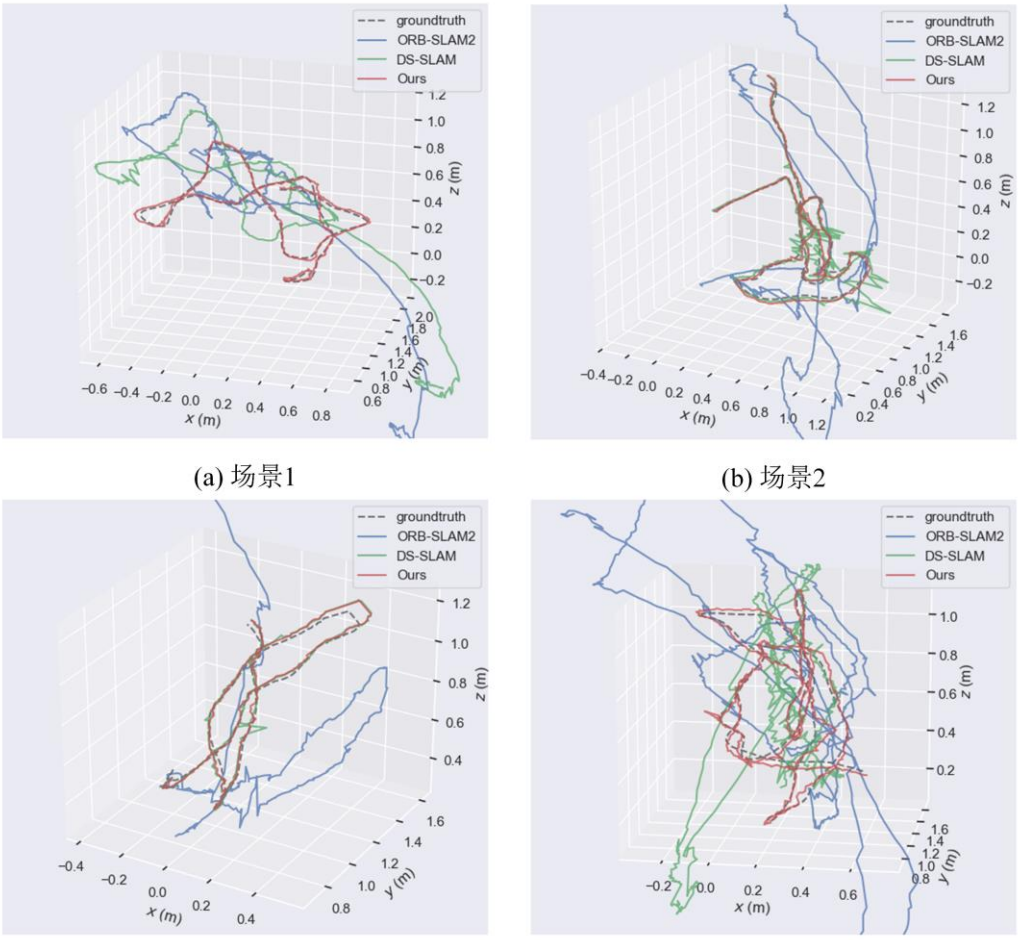
本次实验一共采集 4 组真实场景序列，深度相机 D435i 帧率设定为$30fps$，采集的场景包含无人机起飞到降落的全过程。将真实场景数据转换为标准 TUM 格式后，对 ORB-SLAM2、DS-SLAM 以及本文算法在真实场景数据中的进行实验，然后使用 EVO[95]工具对 ATE 和 RPE 进行评估。ATE 和 RPE 都采用了均方根误差（RMSE）、均值（Mean）、中位数（Median）、标准差（STD）来进行统计。表 5-3 与表 5-4 分别给出了真实场景中三种算法的 ATE 和 RPE，表中“加粗”字体表示运行于其中一个序列的算法（行）在当前评价指标（列）上结果最优，表格带“↓”的四列表示本章算法相较于其他两种算法的误差变化。图 5-9 为三种算法在 4 组动态场景序列的轨迹与真实轨迹的可视化结果。

表 5-3 真实场景实验绝对轨迹误差（ATE）对比

序列	算法	均方根误差	均值	中位数	标准差	均方根误差↓	均值↓	中位数↓	标准差↓
序列 1	ORB-SLAM2	0.6150	0.5253	0.4076	0.3199	96.1%	96.0%	95.3%	96.3%
	DS-SLAM	0.5216	0.4113	0.2722	0.3207	95.4%	94.9%	93.0%	96.3%
	本章算法	**0.0242**	**0.0211**	**0.0190**	**0.0119**	–	–	–	–
序列 2	ORB-SLAM2	0.7618	0.7116	0.6821	0.2719	96.8%	97.0%	97.2%	96.0%
	DS-SLAM	0.0439	0.0312	0.0233	0.0308	45.2%	31.2%	17.2%	65.1%
	本章算法	**0.0240**	**0.0215**	**0.0193**	**0.0108**	–	–	–	–
序列 3	ORB-SLAM2	0.6013	0.4643	0.3200	0.3821	94.1%	93.1%	90.7%	96.0%
	DS-SLAM	0.0355	**0.0317**	**0.0296**	0.0161	0.2%	-1.0%	-0.1%	4.9%
	本章算法	**0.0354**	0.0320	0.0297	**0.0153**	–	–	–	–
序列 4	ORB-SLAM2	0.5954	0.4884	0.3714	0.3406	96.1%	95.7%	94.7%	96.9%
	DS-SLAM	0.4056	0.3299	0.2378	0.2359	94.2%	93.6%	91.7%	95.5%
	本章算法	**0.0235**	**0.0210**	**0.0198**	**0.0106**	–	–	–	–

表 5-4 真实场景实验相对位姿误差（RPE）对比

序列	算法	均方根误差	均值	中位数	标准差	均方根误差↓	均值↓	中位数↓	标准差↓
序列1	ORB-SLAM2	0.0316	0.0255	0.0212	0.0186	36.7%	33.8%	29.8%	42.4%
	DS-SLAM	0.0269	0.0212	0.0167	0.0167	25.7%	20.2%	11.1%	35.6%
	本章算法	**0.0200**	**0.0169**	**0.0149**	**0.0107**	–	–	–	–
序列2	ORB-SLAM2	0.0388	0.0316	0.0253	0.0225	39.0%	35.4%	28.7%	46.8%
	DS-SLAM	0.0399	0.0287	0.0215	0.0276	40.7%	29.1%	16.1%	56.7%
	本章算法	**0.0236**	**0.0204**	**0.0180**	**0.0120**	–	–	–	–
序列3	ORB-SLAM2	0.0430	0.0325	0.0257	0.0281	26.6%	26.8%	20.0%	26.2%
	DS-SLAM	0.0324	**0.0245**	**0.0206**	0.0212	2.5%	2.8%	0.4%	2.2%
	本章算法	**0.0316**	0.0238	0.0205	**0.0208**	–	–	–	–
序列4	ORB-SLAM2	0.0352	0.0284	0.0223	0.0207	38.3%	35.6%	29.3%	43.6%
	DS-SLAM	0.0343	0.0259	0.0199	0.0225	36.7%	29.4%	20.7%	48.0%
	本章算法	**0.0217**	**0.0183**	**0.0158**	**0.0117**	–	–	–	–



(a) 场景1　(b) 场景2

(c) 场景3　(d) 场景4

图 5-9 真实场景三维轨迹可视化

由表5-3可知，本文动态环境下的RGBD SLAM的绝对轨迹误差相比ORB-SLAM2和 DS-SLAM 在真实场景中的大部分评价指标取得了最好的结果，所有评价指标相比ORB-SLAM2 降低了90％以上，RMSE 平均降低95.7％；相比于 DS-SLAM，RMSE 平均降低了58.8％。序列 3 数据录制时间较短，提升幅度不如其他场景，其原因为大部分的 SLAM 算法对短时间的动态场景具有鲁棒性，随着在动态场景中运行的时间变长，针对静态场景设计的 SLAM 算法（本文为 ORB-SLAM2）和部分针对动态场景的SLAM 算法（本文为 DS-SLAM）误差将会不断累积，造成定位精度差，如表中序列 1、序列 2、序列 3 所示。本文算法对长时间、高动态场景鲁棒，因此在长时间的动态场景中优势更为明显。由表 5-4 可知，在真实场景中，本文算法的相对位姿误差所有评价指标优于其余两种算法，RMSE 相比于 ORB-SLAM2 平均降低了35.2％，相比于DS-SLAM 平均降低了26.4％。图 5-9 直观的展示了本文算法相较于 ORB-SLAM2 和DS-SLAM 与真实位姿重合度更好。综上，本文提出的动态环境下 RGBD SLAM 算法在真实复杂动态场景中具有较好的定位精度，能够用于多旋翼无人机在真实动态场景中的自主飞行。

5.3.2 无人机定点悬停实验

多旋翼无人机仅依靠 SLAM 输出的里程计数据能够精确悬停是自主飞行的前提。为了评估本文构建的多旋翼无人机的定点悬停能力和里程计稳定性，设计了实验进行验证。使用无人机自稳模式手动控制无人机在室内运动捕获系统下随机飞行，在飞行中途切换为定点模式，使无人机定位在一特定点处，利用运动捕获系统获取无人机 xyz 轴真实位置，绘制的无人机定点悬停曲线。由于多旋翼无人机的定位精度受电池电量、里程计误差、飞行高度等多方面影响，利用上述方法进行多次定点悬停实验。其中定位误差最大的一组实验设定无人机定位在 $[0.2m, 0.8m, 0.3m]^T$，无人机定点悬停飞行时的实验场景如图 5-10 所示。

图 5-10 无人机定点悬停实验

图 5-11 给出了本次无人机定点悬停实验的位置曲线图，在考虑了多种影响定位精度的条件后，仅依靠视觉里程计进行定位的多旋翼无人机在x轴的最大定位误差为$0.1m$，在y轴的最大定位误差为$0.05m$，在z轴的最大定位误差为$0.09m$，满足本文自主飞行需求。

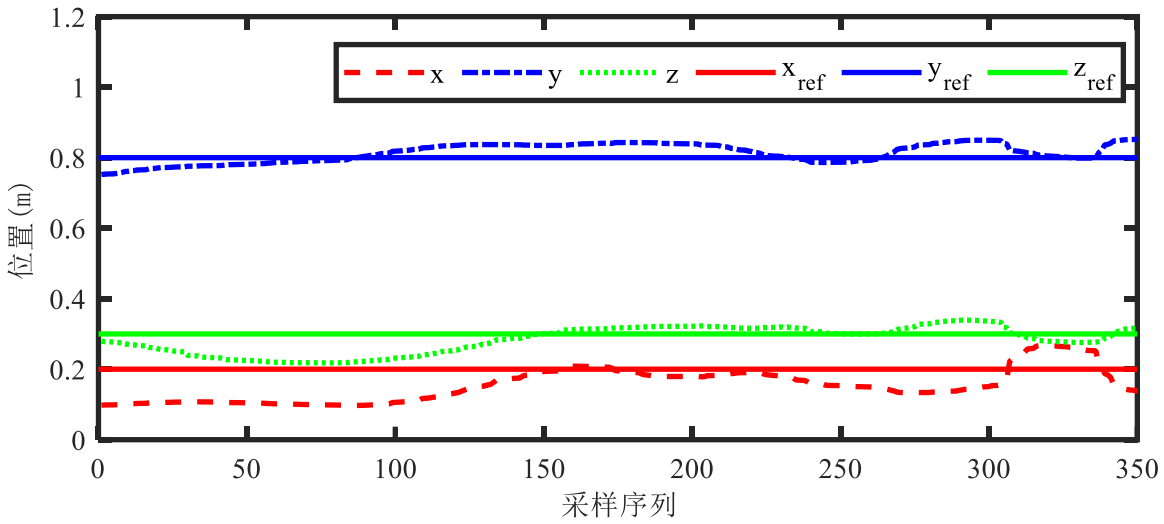


图 5-11 无人机定点悬停位置曲线图

5.3.3　自主飞行验证

本节将验证本文构建的多旋翼无人机自主飞行系统的可行性，分别在室内自行搭建的障碍物环境和室外自然场景进行了实验。

（1） 室内场景实验

室内场景的实验面积约为$7m \times 7m \times 3m$，障碍物由泡沫方块、标定板支架组成。由于室内实验面积有限，无人机最大飞行速度设定为$1m/s$，最大加速度为$5m/s^2$，无人机从起点$[0.0m, -0.2m, 0.6m]^T$安全自主飞行至终点$[3.3m, -0.5m, 1.0m]^T$，室内实验场景和本文算法实时构建的栅格地图如图 5-12 所示。

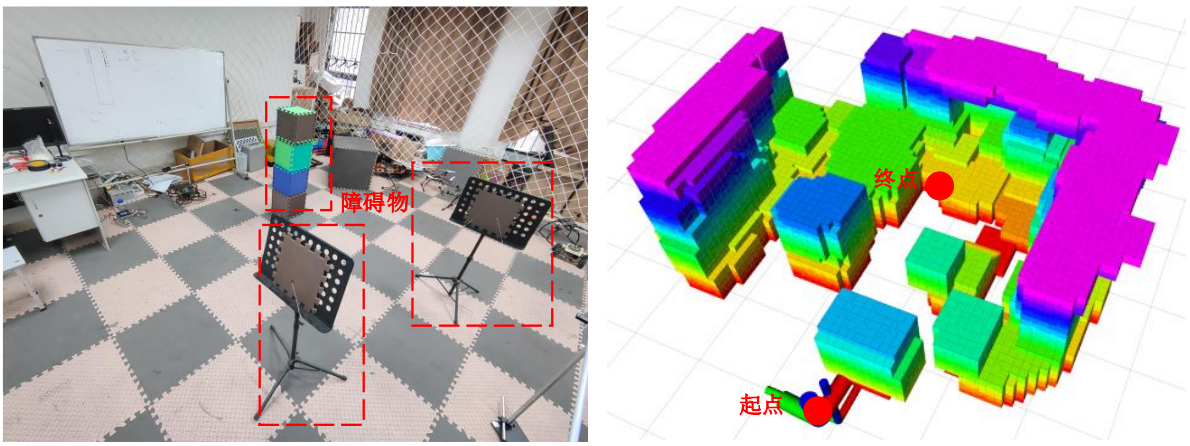


图 5-12 室内实验场景和栅格地图

本文算法将会生成栅格地图，生成避开障碍物且满足无人机动力学的安全飞行轨迹，并控制无人机自主飞行至目标点。其中一次室内场景无人机自主飞行效果如图 5-13 所示。

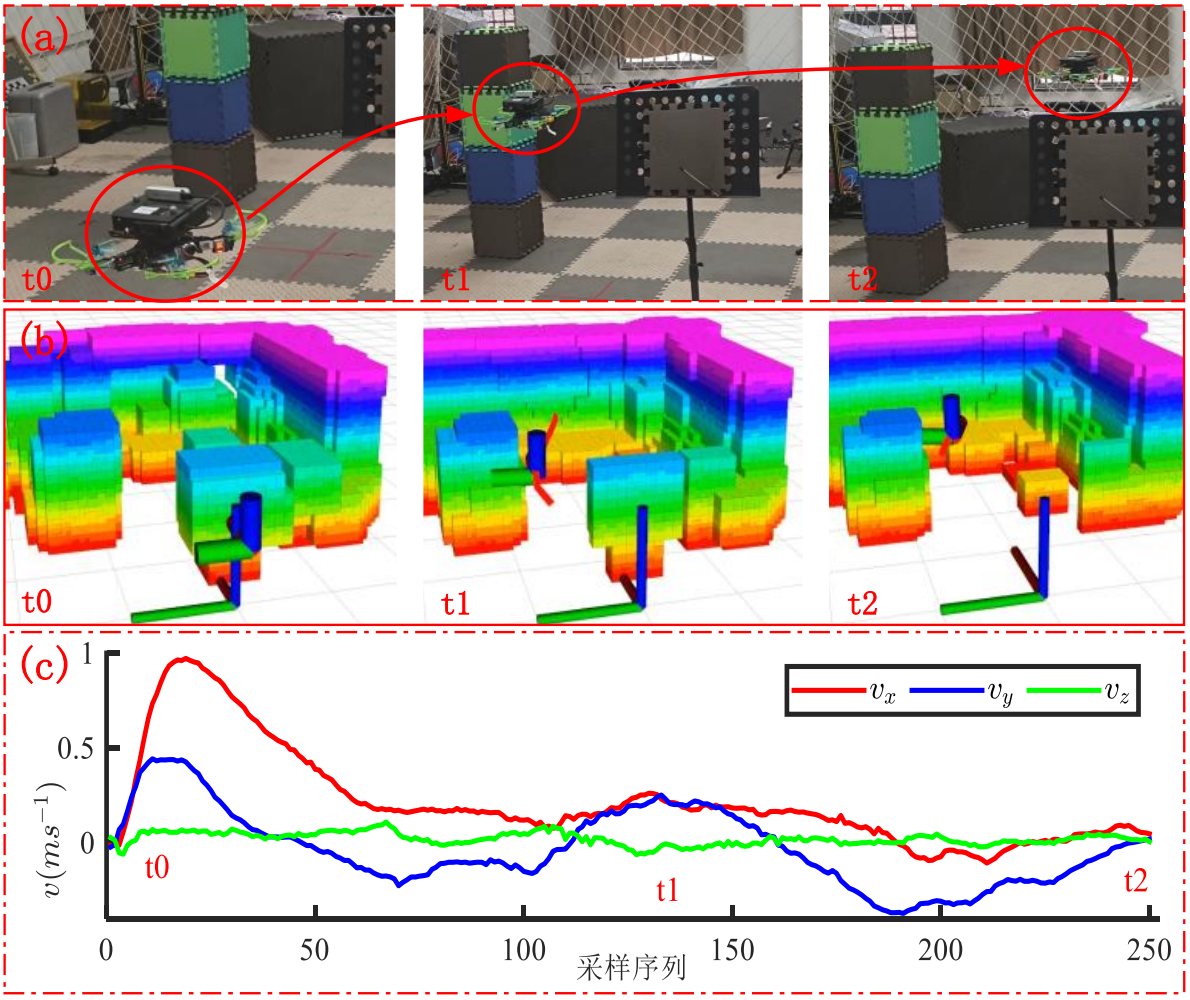


图 5-13 室内场景无人机自主飞行效果图

图 5-13 展示了其中一次室内场景的无人机自主飞行时由起点飞过途经点至终点的过程，包含了不断更新的栅格地图、无人机所处真实位置以及无人机飞行任务途中的各轴速度变化。无人机从t_0时刻出发，绕过正前方的障碍物，在t_1时刻向右飞行，避开左边障碍物，在t_2时刻到达终点，总共花费$6s$。从该过程可以看出，本文算法能够在环境中准确定位、能够生成避开障碍物的安全飞行轨迹、能够控制无人机安全飞往目标点。从 SLAM 算法记录的速度曲线图可以看出，无人机在该室内场景中最大飞行速度可达$0.97m/s$。实验表明，结合了本文第三章、第四章算法的无人机自主飞行系统能够在室内具有静态障碍物的场景中安全稳定飞行。

（2） 室外含动态障碍物场景实验

室外实验场景为校内一处树林，树木密度约为 0.2 棵$/m^2$。由于树木上半部分树枝密集、相互交错，无人机飞行区域为树木底部大约$0\sim2m$区域，如图 5-14 中红色虚

线框所示。该树林中的小路常有人员穿越，同时在无人机飞行的路径中引入额外移动的人体，以验证无人机自主飞行对动态场景的鲁棒性。

图 5-14 室外树林实验场景

在本次实验中，无人机的最大飞行速度设置为$1.5m/s$，最大加速度设定为$6m/s^2$。无人机从悬停在$[0.2m, 0.4m, 1.0m]^T$处自主飞行至终点$[15.0m, 3.0m, 1.0m]^T$无人机构建的地图和飞行轨迹如图 5-15 所示。

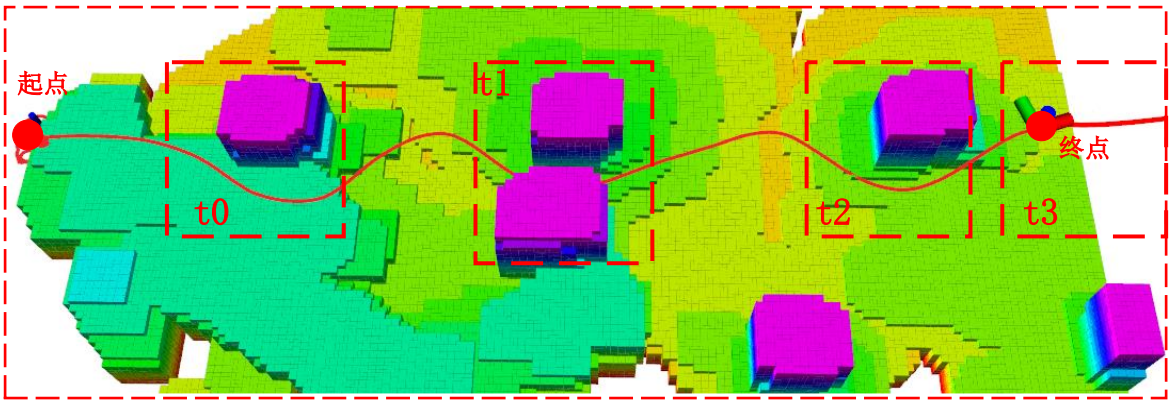


图 5-15 无人机构建的栅格地图和自主飞行轨迹

图 5-15 中在无人机后方的全局栅格地图为地面站记录绘制所得，实际上无人机仅构建深度传感器视场范围内的局部栅格地图。图中无人机在t_0~t_2时刻主动避让了飞行途中的障碍物，并在t_3时到达终点，全程飞行时间为 $20s$。无人机飞行途中t_0~t_3时刻的第三视角、局部栅格地图和飞行速度如图 5-16 所示。

从图 5-16 可以看到，本文无人机系统能够对未知复杂环境进行感知和建图，并控制无人机沿安全飞行轨迹自主飞行。本次实验中，无人机最大前向飞行速度为$1.5m/s$，全程平均飞行速度 $0.7m/s$。在t_1时，无人机飞行途中存在一动态障碍物（人体），无人机依然能够精确定位，并安全避让动态障碍物。实验表明，结合了本文第三章、第四章算法的自主飞行无人机系统能够在室外含动态障碍物场景中安全稳定飞行。

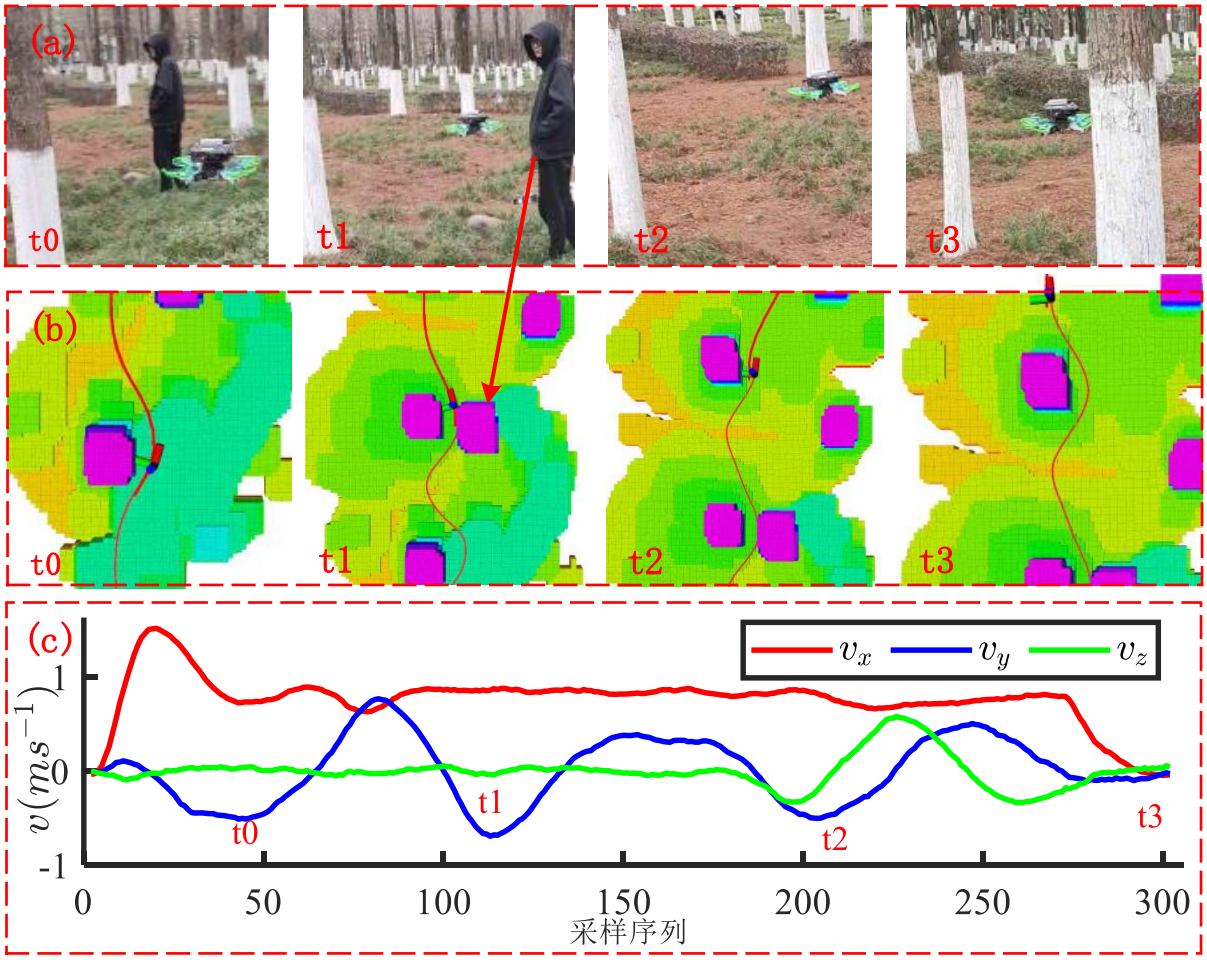


图 5-16 室外静态场景无人机自主飞行效果图

5.4 本章小结

本章结合第三章动态环境 RGBD SLAM 算法和第四章多旋翼无人机运动规划算法，构建了多旋翼无人机自主飞行系统。首先，搭建了系统硬件平台并分析了可行性，其次，对系统中的相机进行了内部参数标定以及对相机与无人机质心进行了外部参数标定。然后，对本文算法框架进行了设计和利用 ROS 机器人操作系统对算法和软件进行了实现。最后，在复杂动态的真实场景中对本文提出的动态环境 RGBD SLAM 算法、无人机定点悬停、无人机自主飞行做了丰富的实验。结果表明，本章构建的多旋翼无人机自主飞行系统在复杂动态的真实场景中能够做到定位准确、以最大$1.5m/s$安全自主飞行。

6 总结与展望

6.1 总结

本文对视觉数据驱动的多旋翼无人机自主飞行关键技术展开研究，主要研究内容和成果简要概括如下：

1. 本文对多旋翼无人机利用视觉在复杂动态场景的定位问题展开了研究，提出了一种基于静态特征点筛选的 RGBD SLAM 算法。首先，利用超轻量目标检测算法PicoDet对场景中的潜在动态对象进行预筛选。然后，对特征点进行光流跟踪并以光流的二范数和方向为特征构建高斯混合模型。针对该高斯混合模型求解子问题，利用期望最大化算法对该模型进行聚类。最后，结合潜在动态对象区域和高斯混合模型聚类结果设计了静态特征点筛选算法，并将其集成到 ORB-SLAM2 算法前端。将本文提出的动态环境 RGBD SLAM、ORB-SLAM2 和 DS-SLAM 在 TUM RGBD 数据集进行对比实验。结果表明，在高动态场景中本文算法的 ATE 均方根误差相比 ORB-SLAM2 平均降低96.0%，相比 DS-SLAM 平均降低了46.2%；本文算法的 RPE 均方根误差相较于 ORB-SLAM2 平均降低了53.4%，相较于 DS-SLAM 平均降低了14.1%。同时，本文算法无需 GPU 参与计算，可在机载计算机上实时运行。

2. 本文对多旋翼无人机在复杂动态环境的运动规划开展了研究，设计了基于样条曲线的多旋翼无人机局部运动规划算法。首先，利用深度相机构建了局部栅格地图减少了构建 ESDF 地图的时间和内存消耗。其次，针对栅格地图不具备 ESDF 地图的方向和梯度信息，设计了障碍物与飞行轨迹之间距离的估计算法，并利用 B 样条对飞行轨迹进行参数化。然后，构建了多种罚函数，利用数值优化方法生成了多旋翼无人机在静态场景和动态场景的安全飞行轨迹。最后，在静态场景、动态场景和不同障碍物密度场景进行了仿真实验，结果表明本文运动规划算法能够在复杂环境中生成多旋翼无人机的安全飞行轨迹。

3. 最后，构建了视觉数据驱动的多旋翼无人机自主飞行实验系统。设计了四旋翼无人机平台，完成了视觉传感器的内外参标定，在 ROS 下实现了本文算法和软件。在真实环境下对该系统进行了实验，结果表明，该系统在复杂环境中定位准确，定点悬停精度为$0.1m$，能够在室内外复杂场景中以$1.5m/s$快速安全飞行。

综上，多种定性和定量的仿真对比和真实场景实验表明，本文提出的基于静态特征点筛选的动态环境 RGBD SLAM 算法能够在复杂动态场景实现鲁棒定位，适用于自主飞行无人机在动态场景精确定位；本文研究的局部运动规划算法能够在复杂动态场景中规划出多旋翼无人机的安全飞行轨迹；本文实现的自主飞行系统能够在静态和动态障碍物环境下安全稳定的自主飞行。

6.2 未来工作展望

本文工作对于视觉数据驱动的多旋翼无人机自主飞行的发展有一定积极意义，但还具有进一步扩展和优化的空间：

（1）文中动态环境下 RGBD SLAM 使用的超轻量目标检测网络是通用的目标检测算法，并没有针对 SLAM 系统专门设计和优化。若将 SLAM 系统中图像的时间上下文信息考虑在内，重新设计专用于 SLAM 的目标检测算法，应能进一步提高潜在动态对象的检测精度和速度。

（2）本文设计的基于简易运动模型的动态障碍物规避算法假设动态障碍物在相机平面内匀速运动，而现实中的动态物体运动状态多种多样，这一假设条件是较为苛刻的，若在动态障碍物规避算法引入障碍物轨迹预测算法，将能进一步提高局部运动规划算法在动态场景的鲁棒性。

（3）本文针对无人机的局部运动规划算法展开研究，无需全局地图信息即可完成无人机自主飞行，但在面对障碍物大小超过无人机感知范围的场景时，无人机运动规划将会失败。若将障碍物的先验信息引入局部运动规划算法，将能降低障碍物大小对运动规划算法的影响。

参考文献

[1] 陶于金, 李沛峰. 无人机系统发展与关键技术综述[J]. 航空制造技术, 2014, 464(20): 34–39.

[2] 全权. 多旋翼飞行器设计与控制[M].电子工业出版社, 2018.

[3] 王成, 杨杰, 姚辉, 等. 四旋翼无人机飞行控制算法综述[J]. 电光与控制, 2018, 25(12): 53–58.

[4] 闫超, 涂良辉, 王聿豪, 等. 无人机在我国民用领域应用综述[J]. 飞行力学, 2022, 40(3): 1–6.

[5] 吴成一. GNSS 拒止条件下的无人机视觉导航研究[D].西安电子科技大学, 2021.

[6] Kazerouni I, Fitzgerald L, Dooly G, et al. A survey of state-of-the-art on visual SLAM [J]. Expert Systems with Applications, 2022, 205: 117734.

[7] Chen W, Shang G, Ji A, et al. An Overview on Visual SLAM: From Tradition to Semantic [J]. Remote Sensing, 2022, 14(13): 3010.

[8] 高兴波, 史旭华, 葛群峰, 等. 面向动态物体场景的视觉 SLAM 综述[J]. 机器人, 2021, 43(6): 733–750.

[9] Jia G, Li X, Zhang D, et al. Visual-SLAM Classical Framework and Key Techniques: A Review [J]. Sensors, 2022, 22(12): 4582.

[10] 牛轶峰, 刘天晴, 李杰, 等. 密集环境中无人机协同机动飞行运动规划方法综述[J]. 国防科技大学学报, 2022, 44(04): 1–12.

[11] Quan L, Han L, Zhou B, et al. Survey of UAV motion planning [J]. IET Cyber-systems and Robotics, 2020, 2(1): 14–21.

[12] Elmokadem T, Savkin A V. Towards fully autonomous UAVs: A survey [J]. Sensors, 2021, 21(18): 6223.

[13] 高翔, 张涛, 刘毅, 等. 视觉 SLAM 十四讲：从理论到实践[M].电子工业出版社, 2017.

[14] Davison A J, Reid I D, Molton N D, et al. MonoSLAM: Real-time single camera SLAM [J]. IEEE transactions on pattern analysis and machine intelligence, 2007, 29(6): 1052–1067.

[15] Klein G, Murray D. Parallel tracking and mapping for small AR workspaces [C]. 2007 6th IEEE and ACM international symposium on mixed and augmented reality. 2007: 225–234.

[16] Mur-Artal R, Montiel J M M, Tardos J D. ORB-SLAM: a versatile and accurate monocular SLAM system [J]. IEEE transactions on robotics, 2015, 31(5): 1147–1163.

[17] Mur-Artal R, Tardós J D. Orb-slam2: An open-source slam system for monocular, stereo, and rgb-d cameras [J]. IEEE transactions on robotics, 2017, 33(5): 1255–1262.

[18] Xiao L, Wang J, Qiu X, et al. Dynamic-SLAM: Semantic monocular visual localization and mapping based on deep learning in dynamic environment [J]. Robotics and Autonomous Systems, 2019, 117: 1–16.

[19] Li A, Wang J, Xu M, et al. DP-SLAM: A visual SLAM with moving probability towards dynamic environments [J]. Information Sciences, 2021, 556: 128–142.

[20] Campos C, Elvira R, Rodríguez J J G, et al. Orb-slam3: An accurate open-source library for visual, visual–inertial, and multimap slam [J]. IEEE Transactions on Robotics, 2021, 37(6): 1874–1890.

[21] Bescos B, Fácil J M, Civera J, et al. DynaSLAM: Tracking, mapping, and inpainting in dynamic scenes [J]. IEEE Robotics and Automation Letters, 2018, 3(4): 4076–4083.

[22] Qin T, Li P, Shen S. VINS-Mono: A Robust and Versatile Monocular Visual-Inertial State Estimator [J]. IEEE Transactions on Robotics, 2018, 34(4): 1004–1020.

[23] Qin T, Cao S, Pan J, et al. A general optimization-based framework for global pose estimation with multiple sensors [J]. arXiv preprint arXiv:1901.03642, 2019.

[24] Qin T, Pan J, Cao S, et al. A general optimization-based framework for local odometry estimation with multiple sensors [J]. arXiv preprint arXiv:1901.03638, 2019.
[25] Yang A J, Cui C, Bârsan I A, et al. Asynchronous multi-view SLAM [C]. 2021 IEEE International Conference on Robotics and Automation (ICRA). 2021: 5669–5676.
[26] Zhang X, Zhong L, Liang C, et al. Metric Calibration of Aerial On-Board Multiple Non-overlapping Cameras Based on Visual and Inertial Measurement Data [C]. Pattern Recognition and Computer Vision. 2021: 16–28.
[27] He K, Gkioxari G, Dollár P, et al. Mask r-cnn [C]. Proceedings of the IEEE international conference on computer vision. 2017: 2961–2969.
[28] Yu C, Liu Z, Liu X-J, et al. DS-SLAM: A Semantic Visual SLAM towards Dynamic Environments [C]. 2018 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS). 2018: 1168–1174.
[29] Liu W, Anguelov D, Erhan D, et al. Ssd: Single shot multibox detector [C]. Computer Vision–ECCV 2016: 14th European Conference, Amsterdam, The Netherlands, October 11–14, 2016, Proceedings, Part I 14. 2016: 21–37.
[30] Schorghuber M, Steininger D, Cabon Y, et al. SLAMANTIC-leveraging semantics to improve VSLAM in dynamic environments [C]. Proceedings of the IEEE/CVF International Conference on Computer Vision Workshops. 2019: 3759–3768.
[31] Bescos B, Campos C, Tardós J D, et al. DynaSLAM II: Tightly-coupled multi-object tracking and SLAM [J]. IEEE robotics and automation letters, 2021, 6(3): 5191–5198.
[32] LaValle S M. Rapidly-exploring random trees: A new tool for path planning [J].1998.
[33] Kavraki L E, Svestka P, Latombe J-C, et al. Probabilistic roadmaps for path planning in high-dimensional configuration spaces [J]. IEEE transactions on Robotics and Automation, 1996, 12(4): 566–580.
[34] Kavraki L E, Kolountzakis M N, Latombe J-C. Analysis of probabilistic roadmaps for path planning [J]. IEEE Transactions on Robotics and automation, 1998, 14(1): 166–171.
[35] Karaman S, Frazzoli E. Sampling-based algorithms for optimal motion planning [J]. The international journal of robotics research, 2011, 30(7): 846–894.
[36] Xinggang W, Cong G, Yibo L. Variable probability based bidirectional RRT algorithm for UAV path planning [C]. The 26th Chinese Control and Decision Conference. 2014: 2217–2222.
[37] Li M, Sun Q, Zhu M. UAV 3-Dimension Flight Path Planning Based on Improved Rapidly-exploring Random Tree [C]. 2019 Chinese Control And Decision Conference (CCDC). 2019: 921–925.
[38] Dijkstra E W. A note on two problems in connexion with graphs [G].Edsger Wybe Dijkstra: His Life, Work, and Legacy. 2022: 287–290.
[39] Hart P E, Nilsson N J, Raphael B. A formal basis for the heuristic determination of minimum cost paths [J]. IEEE transactions on Systems Science and Cybernetics, 1968, 4(2): 100–107.
[40] Dong Z, Chen Z, Zhou R, et al. A hybrid approach of virtual force and A∗ search algorithm for UAV path re-planning [C]. 2011 6th IEEE Conference on Industrial Electronics and Applications. 2011: 1140–1145.
[41] Tianzhu R, Rui Z, Jie X, et al. Three-dimensional path planning of UAV based on an improved A* algorithm [C]. 2016 IEEE Chinese Guidance, Navigation and Control Conference (CGNCC). 2016: 140–145.
[42] Lim D, Park J, Han D, et al. UAV path planning with derivative of the heuristic angle [J]. International Journal of Aeronautical and Space Sciences, 2021, 22: 140–150.
[43] Mellinger D, Kumar V. Minimum snap trajectory generation and control for quadrotors [C]. 2011 IEEE International Conference on Robotics and Automation. 2011: 2520–2525.
[44] Fliess M, Lévine J, Martin P, et al. Flatness and defect of non-linear systems: introductory theory and

examples [J]. International journal of control, 1995, 61(6): 1327–1361.
[45] Richter C, Bry A, Roy N. Polynomial trajectory planning for aggressive quadrotor flight in dense indoor environments [C]. Robotics Research: The 16th International Symposium ISRR. 2016: 649–666.
[46] Chen J, Su K, Shen S. Real-time safe trajectory generation for quadrotor flight in cluttered environments [C]. 2015 IEEE International Conference on Robotics and Biomimetics (ROBIO). 2015: 1678–1685.
[47] Gao F, Shen S. Online quadrotor trajectory generation and autonomous navigation on point clouds [C]. 2016 IEEE International Symposium on Safety, Security, and Rescue Robotics (SSRR). 2016: 139–146.
[48] Gao F, Wu W, Lin Y, et al. Online safe trajectory generation for quadrotors using fast marching method and bernstein basis polynomial [C]. 2018 IEEE International Conference on Robotics and Automation (ICRA). 2018: 344–351.
[49] Oleynikova H, Millane A, Taylor Z, et al. Signed distance fields: A natural representation for both mapping and planning [C]. RSS 2016 Workshop: Geometry and Beyond-Representations, Physics, and Scene Understanding for Robotics. 2016.
[50] Zhou B, Gao F, Pan J, et al. Robust real-time uav replanning using guided gradient-based optimization and topological paths [C]. 2020 IEEE International Conference on Robotics and Automation (ICRA). 2020: 1208–1214.
[51] Zhou X, Wang Z, Ye H, et al. EGO-Planner: An ESDF-Free Gradient-Based Local Planner for Quadrotors [J]. IEEE Robotics and Automation Letters, 2021, 6(2): 478–485.
[52] Zhou X, Zhu J, Zhou H, et al. EGO-Swarm: A Fully Autonomous and Decentralized Quadrotor Swarm System in Cluttered Environments [C]. 2021 IEEE International Conference on Robotics and Automation (ICRA). 2021: 4101–4107.
[53] Zhou X, Wen X, Wang Z, et al. Swarm of micro flying robots in the wild [J]. Science Robotics, 2022, 7(66): eabm5954.
[54] Tordesillas J, How J P. PANTHER: Perception-Aware Trajectory Planner in Dynamic Environments [J]. IEEE Access, 2022, 10: 22662–22677.
[55] Meier L. Pixhawk [EB/OL]. (2023) .https://github.com/pixhawk/Hardware.
[56] Rublee E, Rabaud V, Konolige K, et al. ORB: An efficient alternative to SIFT or SURF [C]. 2011 International conference on computer vision. 2011: 2564–2571.
[57] Ng P C, Henikoff S. SIFT: Predicting amino acid changes that affect protein function [J]. Nucleic acids research, 2003, 31(13): 3812–3814.
[58] Viswanathan D G. Features from accelerated segment test (fast) [C]. Proceedings of the 10th workshop on image analysis for multimedia interactive services. 2009: 6–8.
[59] Muja M, Lowe D G. Fast approximate nearest neighbors with automatic algorithm configuration. [J]. VISAPP, 2009, 2(2): 331–340.
[60] 高翔. 机器人学中的状态估计[M].西安交通大学出版社, 2018.
[61] Hartley R, Zisserman A. Multiple view geometry in computer vision [M].Cambridge university press, 2003.
[62] Wu Y, Hu Z. PnP problem revisited [J]. Journal of Mathematical Imaging and Vision, 2006, 24(1): 131–141.
[63] Triggs B, McLauchlan P F, Hartley R I, et al. Bundle adjustment—a modern synthesis [C]. International workshop on vision algorithms. 1999: 298–372.
[64] Schubert D, Goll T, Demmel N, et al. The TUM VI Benchmark for Evaluating Visual-Inertial Odometry [C]. 2018 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS). 2018: 1680–1687.

[65] Lozano-Perez T. Spatial planning: A configuration space approach [M].1990.
[66] Oleynikova H, Taylor Z, Fehr M, et al. Voxblox: Incremental 3D Euclidean Signed Distance Fields for on-board MAV planning [C]. 2017 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS). 2017: 1366–1373.
[67] Han L, Gao F, Zhou B, et al. FIESTA: Fast Incremental Euclidean Distance Fields for Online Motion Planning of Aerial Robots [C]. 2019 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS). 2019: 4423–4430.
[68] Pan Y, Kompis Y, Bartolomei L, et al. Voxfield: Non-Projective Signed Distance Fields for Online Planning and 3D Reconstruction [C]. IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS). 2022: 5331–5338.
[69] Barath D, Cavalli L, Pollefeys M. Learning To Find Good Models in RANSAC [C]. Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition. 2022: 15744–15753.
[70] Lee D-S. Effective Gaussian mixture learning for video background subtraction [J]. IEEE transactions on pattern analysis and machine intelligence, 2005, 27(5): 827–832.
[71] Moon T K. The expectation-maximization algorithm [J]. IEEE Signal processing magazine, 1996, 13(6): 47–60.
[72] Yu G, Chang Q, Lv W, et al. PP-PicoDet: A Better Real-Time Object Detector on Mobile Devices [J]. arXiv preprint arXiv:2111.00902, 2021.
[73] Ma N, Zhang X, Zheng H-T, et al. Shufflenet v2: Practical guidelines for efficient cnn architecture design [C]. Proceedings of the European conference on computer vision (ECCV). 2018: 116–131.
[74] Bochkovskiy A, Wang C-Y, Liao H-Y M. Yolov4: Optimal speed and accuracy of object detection [J]. arXiv preprint arXiv:2004.10934, 2020.
[75] Ge Z, Liu S, Wang F, et al. Yolox: Exceeding yolo series in 2021 [J]. arXiv preprint arXiv:2107.08430, 2021.
[76] Liu S, Qi L, Qin H, et al. Path aggregation network for instance segmentation [C]. Proceedings of the IEEE conference on computer vision and pattern recognition. 2018: 8759–8768.
[77] Howard A, Sandler M, Chu G, et al. Searching for mobilenetv3 [C]. Proceedings of the IEEE/CVF international conference on computer vision. 2019: 1314–1324.
[78] Hu J, Shen L, Sun G. Squeeze-and-excitation networks [C]. Proceedings of the IEEE conference on computer vision and pattern recognition. 2018: 7132–7141.
[79] Han K, Wang Y, Tian Q, et al. Ghostnet: More features from cheap operations [C]. Proceedings of the IEEE/CVF conference on computer vision and pattern recognition. 2020: 1580–1589.
[80] Kumar A, Vashishtha G, Gandhi C P, et al. Novel convolutional neural network (NCNN) for the diagnosis of bearing defects in rotary machinery [J]. IEEE Transactions on Instrumentation and Measurement, 2021, 70: 1–10.
[81] Lucas B D, Kanade T, others. An iterative image registration technique with an application to stereo vision [C]. IJCAI'81: 7th international joint conference on Artificial intelligence. 1981, 2: 674–679.
[82] 李婧. 基于 GMM 的 EM 优化算法的应用与研究[D].哈尔滨工程大学, 2018.
[83] Ahmed M, Seraj R, Islam S M S. The k-means Algorithm: A Comprehensive Survey and Performance Evaluation [J]. Electronics, 2020, 9(8): 1295.
[84] LaValle S M. Planning algorithms [M].Cambridge university press, 2006.
[85] Usenko V, von Stumberg L, Pangercic A, et al. Real-time trajectory replanning for MAVs using uniform B-splines and a 3D circular buffer [C]. 2017 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS). 2017: 215–222.
[86] Liu D C, Nocedal J. On the limited memory BFGS method for large scale optimization [J]. Mathematical programming, 1989, 45(1): 503–528.
[87] Zhou B, Pan J, Gao F, et al. Raptor: Robust and perception-aware trajectory replanning for quadrotor

fast flight [J]. IEEE Transactions on Robotics, 2021, 37(6): 1992–2009.
[88] Wu W. Mockamap [EB/OL]. (2022) .https://github.com/HKUST-Aerial-Robotics/mockamap.
[89] Zhou B, Gao F, Wang L, et al. Robust and efficient quadrotor trajectory generation for fast autonomous flight [J]. IEEE Robotics and Automation Letters, 2019, 4(4): 3529–3536.
[90] Shi D, Dai X, Zhang X, et al. A practical performance evaluation method for electric multicopters [J]. IEEE/ASME Transactions on Mechatronics, 2017, 22(3): 1337–1348.
[91] Zhang Z. A flexible new technique for camera calibration [J]. IEEE Transactions on pattern analysis and machine intelligence, 2000, 22(11): 1330–1334.
[92] Olson E. AprilTag: A robust and flexible visual fiducial system [C]. 2011 IEEE international conference on robotics and automation. 2011: 3400–3407.
[93] Furgale P, Rehder J, Siegwart R. Unified temporal and spatial calibration for multi-sensor systems [C]. 2013 IEEE/RSJ International Conference on Intelligent Robots and Systems. 2013: 1280–1286.
[94] Furtado J S, Liu H H, Lai G, et al. Comparative analysis of optitrack motion capture systems [C]. Advances in Motion Sensing and Control for Robotic Applications: Selected Papers from the Symposium on Mechatronics, Robotics, and Control (SMRC'18)-CSME International Congress. 2019: 15–31.
[95] Grupp M. evo: Python package for the evaluation of odometry and SLAM. [EB/OL]. (2017) .https://github.com/MichaelGrupp/evo.