http://bhxb.buaa.edu.cn jbuaa@buaa.edu.cn
DOI: 10.13700/j.bh.1001-5965.2024.0658

一种复杂动态场景实时视觉SLAM算法

赵舶廷[1]，刘军[1,*]，柴宏旭[2]，张健烨[1]，吴睿彬[1]

(1. 兰州理工大学 机电工程学院，兰州 730050；2. 甘肃省特种设备检验检测研究院，兰州 730050)

摘 要：针对存在动态物体的一般性或复杂动态场景下同步定位与地图构建（SLAM）系统往往存在强假设性条件或定位精度不足乃至失效等多种问题，提出一种复杂动态场景实时视觉SLAM算法。该算法以ORB-SLAM3为基础，部署基于加速模型推理的YOLOX-s获取语义先验信息以处理动态目标，并结合卡尔曼滤波和匈牙利算法补偿目标检测缺失；为提升算法定位精度和泛化性能，在利用基于网格的运动统计（GMS）特征匹配算法快速筛选有效匹配点的基础上，提出一种双类别动态场景处理策略，根据图像匹配精度将场景分为两类处理：对低动态帧，采用GMS算法快速过滤动态特征，避免进行大量矩阵运算从而提升运行效率；对高动态帧，在利用卡方检验识别运动异常值的同时，提出一种动态特征剔除策略，根据自适应阈值剔除动态点并保持跟踪稳定性。标准数据集与真实场景下的实验测试均表明，所提算法能解决室内外复杂动态场景下的实时定位问题，表现出良好的精确性、鲁棒性和及时性。

关 键 词：视觉SLAM；动态物体；深度学习；目标检测；场景分类

中图分类号：TP242.6

文献标志码：A **文章编号**：1001-5965(2026)11-4075-15

同步定位与地图构建(simultaneous localization and mapping, SLAM)是智能移动机器人环境感知和自主定位的关键技术之一，其是指机器人利用特定的传感器模组，在未知环境中估计自身运动，于运动过程中构建环境的模型[1]。自20世纪80年代首次提出该问题以来，经过数十年的研究，SLAM技术取得了显著进展，由于摄像机等视觉传感器具有成本低、信息量丰富、使用场景多样的特点，视觉SLAM具有巨大的发展潜力和广阔的应用前景。

前人一般基于静态场景开展相关SLAM技术研究，即假设环境中所有物体为静态刚体。但日常生活和工业应用等现实环境中普遍存在动态物体，其引入的误导性数据关联会影响相机的跟踪精度和生成地图的可靠性。

对于环境中的动态物体，普遍采用的解决方案是：从SLAM的输入数据中识别动态成分，将其视为异常值丢弃以避免其干扰相机位姿估计和地图构建。例如，基于定向FAST角点与旋转BRIEF描述子的同时定位与地图构建[2](oriented FAST and rotated BRIEF-simultaneous localization and mapping, ORB-SLAM)利用随机采样一致算法过滤外点，并行跟踪与地图构建[3](parallel tracking and mapping, PTAM)则使用鲁棒核函数处理异常值。也有一些算法在经典SLAM框架的基础上，为静态特征设置一个带几何约束的阈值，利用阈值检测动态特征，再将其视作离群值加以移除以应对场景中的动态物体。除此之外，各国学者围绕该问题也开展了一些其他的努力与尝试，Cheng等[4]通过比较相邻两帧图像之间的相似性和差异性，基于贝叶斯框架将场景划分为动态和静态区域，但由于其采用固定的阈值进

收稿日期：2024-09-05；录用日期：2024-11-07；网络出版时间：2024-12-03 12:14
网络出版地址：link.cnki.net/urlid/11.2625.V.20241203.0904.002
基金项目：国家自然科学基金(71861025)；甘肃省科技厅自主研发计划-工业类(22YF11GA315)；兰州市科技支撑专项（2024-8-46）
*通信作者. E-mail：lzhjliu@126.com

引用格式：赵舶廷，刘军，柴宏旭，等. 一种复杂动态场景实时视觉SLAM算法[J]. 北京航空航天大学学报，2026，52（11）：4075-4089.
Zhao B T，Liu J，Chai H X，et al. A real-time visual SLAM algorithm for complex dynamic scenarios[J]. Journal of Beijing University of Aeronautics and Astronautics，2026，52（11）：4075-4089（in Chinese）.

行区域划分，在低动态场景中常存在区域划分错误的情况；Dai等[5]使用Delaunay三角剖分法，利用特征点之间的相关性来分离动态点和静态点，但不适用于运动物体占比过大的场景；Du等[6]使用具有长期一致性的条件随机场，通过整合多帧长期观测信息来检测动态成分，但也存在静态场景跟踪失败的问题。总体而言，上述方法一般不依赖先验信息，部分方法中相关固定阈值主要基于经验进行确定，并且只能处理简单环境或动态物体较少的场景，当动态物体所占图像比例过大、数量较多或存在遮挡等复杂场景时，上述方法存在失效问题。

为此，部分学者近年来将神经网络集成至SLAM系统中，尝试利用深度学习方法解决该问题。其主要思路是利用神经网络处理语义先验信息来划分潜在动态区域，并进而结合几何约束消除动态特征对SLAM的影响。

例如，一些算法将语义分割网络和SLAM系统相结合，Bescos等[7]使用Mask R-CNN对先验动态物体进行像素级实例分割，再结合多视图几何方法剔除动态物体上的特征点；Yu等[8]使用轻量化网络SegNet获取语义掩膜，通过计算帧间匹配点的极线距离去除离群值；王浩等[9]结合实例分割网络YOLACT与深度图像聚类的结果，通过实例分割修补算法和基于实例级对象的运动状态判定算法，优化动态环境下的相机位姿估计。上述基于语义分割的方法可获得像素级掩膜，但精细分割导致计算成本过高，逐帧处理图像的方式往往难以保障实时性，不能满足在线处理的要求，而且掩膜边界的划分结果也并不完全可靠。

也有部分学者将目标检测网络作为附加线程引入到SLAM系统中，例如，齐浩等[10]采用轻量化网络YOLOv7-tiny结合光流检测动态区域，并基于图像对比度自适应调整特征点提取数量以应对光照变化对特征提取的不良影响；Soares等[11]先基于YOLOV3训练得到网络模型CYTi，再根据检测结果移除冗余特征点，但其不能有效识别人类以外的其他运动物体；Cheng等[12]创建了2个用于检测运动目标和构建语义地图的新线程，并提出了一个融合语义和几何信息的快速动态特征剔除算法；He等[13]在优化环节中，赋予每个特征点一个优化权重，降低潜在动态特征对位姿估计的不利影响。虽然结合目标检测的SLAM算法一般比语义分割的SLAM算法更加高效，但由于无法获取完整的物体边缘信息，检测框内可能包含其他背景信息，若简单移除框内所有特征点可能会导致算法失效，仍需额外制定对应处理策略；并且当2个检测框存在交集时可能会产生歧义；此外，若将所有动态特征视作同一属性，处理多变的现实场景时也存在算法泛化能力不足的问题。

总体而言，虽然当前基于神经网络的方法在不同程度上提高了部分动态环境下相关SLAM算法的跟踪精度，但均存在效率和性能难以兼顾的问题，各国学者仍在不断努力以满足不同应用场景下的技术需求。此外，该类方法大多具有一个强假设条件，即限定动态物体为某些特定类，如生活环境中的人类和车辆，但当出现未知动态物体时，其可靠性会大幅下降。

基于此，本文针对动态物体数量较多、运动速度、幅度差异性大或者物体间存在遮挡的复杂动态场景，提出一种实时语义动态视觉SLAM算法。该算法在ORB-SLAM3的框架上进行扩展，集成轻量化网络YOLOX-s获取目标对象的语义先验信息，假设所有被检测到的目标都可能处于运动状态；利用卡尔曼滤波与匈牙利算法预测目标的运动轨迹以补偿目标检测中可能出现的遗漏和缺失；通过在匹配优化中GMS算法得到的匹配结果计算匹配精度，再根据匹配精度将动态场景细分为两类进行针对性处理，在兼顾实时性应用需求的同时，有效提升算法定位精度；在此基础之上，结合静态目标的语义信息进行场景重建，构建具有全局一致性的静态稠密语义点云地图。

1 SLAM算法实现

鉴于ORB-SLAM3[14]在定位精度和鲁棒性方面优于同类基于特征点的视觉SLAM算法，且支持单目，双目和RGB-D等多种相机，选择以ORB-SLAM3为基础框架进行二次开发。

本文算法框架如图1所示，为实时、准确获取目标对象的语义类别标签和检测框的像素坐标，在ORB-SLAM3前端部署利用TensorRT (https://developer.nvidia.com/tensorrt)加速模型推理的轻量化网络YOLOX-s[15]进行目标检测。针对待检目标存在检测遗漏、缺失的问题，利用卡尔曼滤波预测目标运动轨迹，匈牙利算法建立数据关联，对目标检测结果进行补偿。

同时，在最小化运行时间成本的条件下，为提高不同动态程度场景下SLAM系统的定位精度，在图像特征匹配环节，首先利用GMS特征匹配过滤算法[16]快速筛选得到有效匹配点，在此基础上，设计一种基于匹配精度的双类别动态场景处理策略，根据帧间匹配精度将图像帧分为高低动态2个类别处理：对低动态场景，将GMS算法得到的匹配点

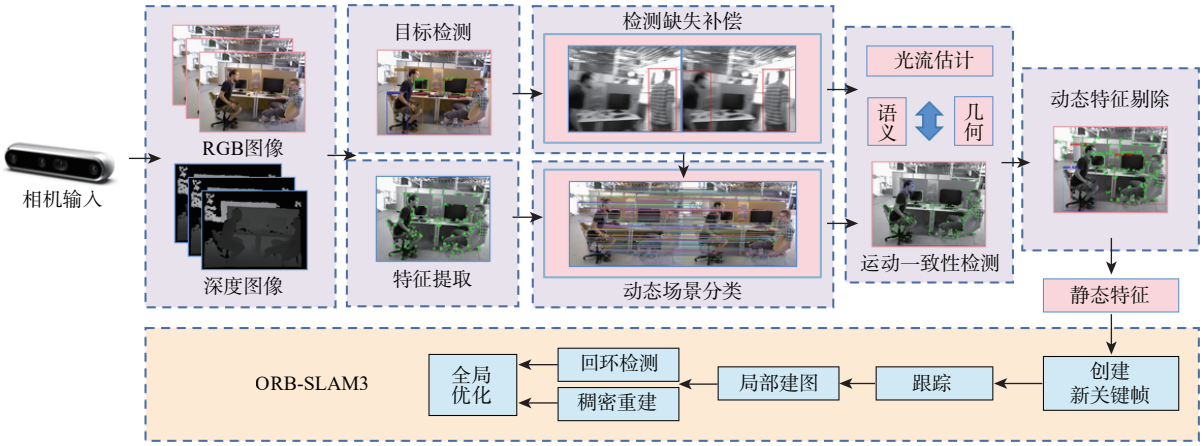


图 1　SLAM 算法框架

Fig. 1　Frame of SLAM algorithm

直接用于相机位姿估计，快速鲁棒地过滤轻微的动态特征；对高动态场景，结合光流和对极几何约束检测动态异常值，考虑到固定的经验阈值对未知环境的泛化性能不足，引入卡方检验识别检测框内点的运动状态，再结合语义先验移除离群值。

此外，本文提出一种动态特征剔除策略，根据检测框内特征点总数自适应调整阈值确定动态框，当检测框间存在重叠时，保留静态框内点和框外背景点，而不是简单地消除动态框内所有点以增加稳定的静态特征点数量，从而提升 SLAM 系统的跟踪稳定性。

最后，当以 RGB-D 相机作为输入时，将二维图像中静态目标的语义信息投影至三维点云，经过点云滤波、拼接和染色处理，构建具有全局一致性的静态语义点云地图，有助于机器人开展导航避障、识别抓取和人机交互等后续工作。

1.1　目标检测缺失补偿

有效检测和识别动态物体并针对性加以处理是提升动态场景下 SLAM 算法定位精度的关键之一。为保障及时性要求，本文采用推理效率相对较高但结构简单的 YOLOX-s 网络进行目标检测，其目的是利用神经网络获取目标对象的类别和位置信息，为后续处理策略提供语义先验。YOLOX-s 能够同时识别 80 类对象，基本满足现实场景的应用需要。并且，为进一步提高推理效率，本文亦采用模型推理优化器 TensorRT 从层和张量融合、内核自动调整、多流并行等角度，对 YOLOX-s 的网络结构进行重构和优化。

虽然轻量化目标检测网络推理效率高，但其模型参数和计算量有限、特征提取能力较弱，可能因运动模糊、视角转换、目标遮挡、视角内目标不完整等因素的干扰而失效，尤其是环境中存在动态目标，语义先验的缺失会影响图像特征提取并降低匹配精度，同时产生大量错误的数据关联，进而影响 SLAM 系统的跟踪和定位。对动态目标进行跟踪，利用跟踪结果补偿检测缺失可以在一定程度上弥补这种缺陷。

考虑到干扰因素可能导致相机输入图像质量下降，从而降低目标检测结果的置信度甚至出现检测缺失，本文将 YOLOX-s 的检测置信度阈值设为 0，而非常见的 0.4 或 0.5，即保留所有置信度大于 0 的检测框进行处理。在保证检测效率的基础上，最大限度地降低首帧中运动目标检测缺失的不良影响。

由于相机的采样频率通常在 30 Hz 以上，相邻图像帧间运动目标的移动距离较小，满足卡尔曼滤波的恒速运动模型假设，本文采用多目标跟踪算法 BoT-SORT[17] 对场景中的人类和车辆等潜在运动目标进行跟踪，对 YOLOX-s 的检测缺失进行补偿。

图 2 为本文使用的多目标跟踪算法流程，其中，S 为置信度。首先输入当前帧中潜在运动目标的检测结果 $\boldsymbol{D}_t$，包括检测框的置信度 θ、语义类别及图像平面上的二维坐标。再利用 OpenCV 库集成的相机全局运动补偿技术对上一帧的跟踪轨迹 $\boldsymbol{T}_{t-1}$ 进行校正，针对卡尔曼滤波只根据检测框宽高比进行轨迹预测，导致预测结果不佳的问题，直接对检测框的宽和高分别进行预测，获得更为准确的预测结果。通过卡尔曼滤波器对潜在运动目标进行轨迹预测，获得的结果 $\boldsymbol{T}_{\mathrm{p}}$ 可表示为

$$\boldsymbol{T}_{\mathrm{p}}=\begin{cases}\hat{\boldsymbol{x}}_t^{-}=\boldsymbol{F}_t\hat{\boldsymbol{x}}_{t-1}\\ \boldsymbol{P}_t^{-}=\boldsymbol{F}_{t-1}\boldsymbol{P}_{t-1}+\boldsymbol{Q}_t\end{cases}\tag{1}$$

输入为当前帧的系统噪声协方差矩阵 $\boldsymbol{Q}_t$，上一帧的跟踪轨迹状态转移矩阵 $\boldsymbol{F}_{t-1}$、状态向量 $\hat{\boldsymbol{x}}_{t-1}$ 和状态协方差矩阵 $\boldsymbol{P}_{t-1}$，输出为当前帧的预测轨迹状态向量 $\hat{\boldsymbol{x}}_t^{-}$ 和状态协方差矩阵 $\boldsymbol{P}_t^{-}$，预测轨迹状态向量

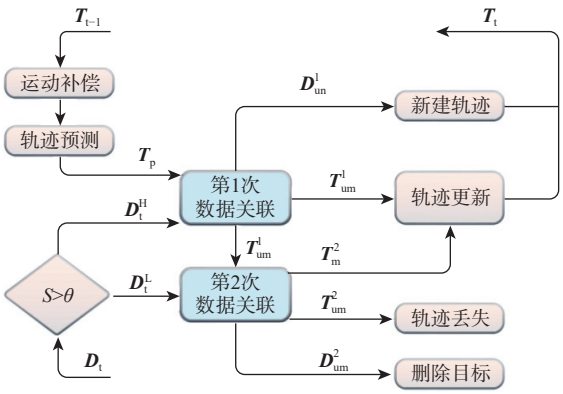


图2 多目标跟踪算法流程

Fig. 2 Multi object tracking algorithm flow chart

$\hat{\boldsymbol{x}}_t^-$可表示为

$$\hat{\boldsymbol{x}}_t^- = [x_c, y_c, w, h, x_c', y_c', w', h']^{\mathrm{T}} \tag{2}$$

式中：(x_c, y_c)为检测框的中心坐标；w和h分别为宽和高；x_c'、y_c'、w'、h'表示各项变化速率。

同时，将当前帧的检测结果$\boldsymbol{D}_t$分成低置信度$\boldsymbol{D}_t^{\mathrm{L}}$和高置信度$\boldsymbol{D}_t^{\mathrm{H}}$两类，为保持SLAM系统的整体运行效率，由式(3)计算$\boldsymbol{D}_t^{\mathrm{H}}$中的检测框$D_i$和预测框$P_i$的交并比(intersection of union, IoU)：

$$i(D_i, P_i) = \frac{D_i \cap P_i}{D_i \cup P_i} \tag{3}$$

使用交并比$i(D_i, P_i)$表示匹配相似度，若$i(D_i, P_i)$小于设定阈值，则表示预测框不能与YOLOX-s获取的检测框建立匹配关系，即当前帧中出现目标检测缺失，使用预测框作为当前帧的目标检测结果对检测缺失进行补偿。

数据关联则是通过计算一个$N \times M$代价矩阵实现，其中，N表示检测框的数量，M表示预测框的数量，代价矩阵$\boldsymbol{I}(\boldsymbol{D}, \boldsymbol{P})$的每个元素如下：

$$\boldsymbol{I}(\boldsymbol{D}, \boldsymbol{P}) = \begin{bmatrix} i(D_1, P_1) & \cdots & i(D_1, P_M) \\ i(D_2, P_1) & \cdots & i(D_2, P_M) \\ \vdots & & \vdots \\ i(D_N, P_1) & \cdots & i(D_N, P_M) \end{bmatrix} \tag{4}$$

匈牙利算法利用代价矩阵$\boldsymbol{I}(\boldsymbol{D}, \boldsymbol{P})$对检测框和预测框进行匹配，将其转化为最优任务分配问题，使得当前帧总体匹配相似度最大，进而确定预测框与检测框的对应关系，实现对潜在运动目标的连续跟踪。此外，根据匹配相似度建立代价矩阵进行2次数据关联，具有计算量小、可靠性高的优点，可实现算法运行效率与精度的均衡。

匹配失败的轨迹$\boldsymbol{T}_{\mathrm{um}}^1$可与$\boldsymbol{D}_t^{\mathrm{H}}$再次进行数据关联，删除匹配失败的轨迹$\boldsymbol{T}_{\mathrm{um}}^2$和低分检测框$\boldsymbol{D}_{\mathrm{um}}^{\mathrm{L}}$，再将有效匹配轨迹$\boldsymbol{T}_{\mathrm{m}}^1$和$\boldsymbol{T}_{\mathrm{m}}^2$输入卡尔曼滤波器进行轨迹更新，其更新公式为

$$\boldsymbol{T}_{\mathrm{u}} = \begin{cases} \hat{\boldsymbol{x}}_t = \hat{\boldsymbol{x}}_t^- + K_t(\boldsymbol{z}_t - \boldsymbol{H}_t\hat{\boldsymbol{x}}_t^-) \\ K_t = \boldsymbol{P}_t^- \boldsymbol{H}_t^{\mathrm{T}}(\boldsymbol{H}_t \boldsymbol{P}_t^- \boldsymbol{H}_t^{\mathrm{T}} + \boldsymbol{R}_t)^{-1} \\ \boldsymbol{P}_t = \boldsymbol{P}_t^- - \boldsymbol{P}_t^- K_t \boldsymbol{H}_t \end{cases} \tag{5}$$

式中：K_t为卡尔曼增益权重；$\boldsymbol{H}_t$为尺度转换矩阵；$\boldsymbol{R}_t$为检测噪声协方差矩阵；$\boldsymbol{z}_t$为当前帧的观测向量；输出为轨迹更新后的状态向量$\hat{\boldsymbol{x}}_t$和状态协方差矩阵为$\boldsymbol{P}_t$。

对于未匹配成功的高分检测框$\boldsymbol{D}_{\mathrm{um}}^{\mathrm{H}}$，为其新建一个轨迹，最终得到当前帧的全部跟踪轨迹$\boldsymbol{T}_t$。

图3为上述4种干扰因素和连续三帧检测缺失的检测补偿结果。其中，图3(a)为YOLOX-s的目标检测结果，图3(b)则是引入检测缺失补偿后的检测结果。对比图3(a)和图3(b)可以看出：通过对前后帧之间目标的运动状态进行连续估计，本文算法有效弥补了目标检测中存在的缺失和遗漏，可以更为准确、高效地获取目标对象的位置信息。



(a) 目标检测结果



(b) 检测补偿结果

图3 检测缺失补偿结果

Fig. 3 Results of missed detection compensation

1.2 双类别动态场景处理策略

对目标对象进行有效检测与跟踪，进而获得准确的语义先验信息，再根据特定的语义类别就能够识别出潜在运动目标。然而，现有结合语义信息和几何信息的SLAM算法处理动态目标时大多需要求解大量基础矩阵，算法运行效率难以保证。此

外, 其认为场景内所有运动物体都处于快速或大幅度运动状态, 对环境中微小的运动变化并不敏感, 导致算法在处理不同运动程度的物体时泛化性能不足。

针对此类问题, 本文提出一种基于特征匹配精度的双类别动态场景处理策略, 在图像特征匹配环节, 首先使用 GMS 算法快速筛选得到有效的匹配特征点, 再结合目标检测获得的语义信息计算图像帧间特征匹配精度, 并以此为依据划分该帧的动态程度, 将场景分成高动态和低动态两类。在低动态场景中, 直接使用 GMS 算法过滤异常点, 将筛选得到的匹配点用于相机位姿估计, 该方法不涉及复杂的矩阵分解, 运行速度更快; 在高动态场景中, 通过光流估计结合对极几何约束识别动态点, 再设计对应策略对其进行剔除。

对动态程度不同的场景加以区分, 再分别做针对性处理, 能够节约大量运行时间成本, 同时提升 SLAM 系统在复杂动态场景下的跟踪精度。

1.2.1　特征匹配筛选

GMS 算法遵循运动平滑假设, 即正确匹配的领域内具有运动一致性约束, 如果一个特征匹配点对匹配正确, 那么其周围还有其他特征匹配点对与之形成相互支持。借由像素网格划分候选区域, 统计运动支持度得分, 根据正误匹配概率分布的可分性, 设定阈值判定匹配的精确度, 再根据动态区域和静态区域的运动支持度得分不同, 可有效识别和处理场景中有限的动态特征[18]。

提取 ORB 特征点完毕后, 本文使用 Brute Force 算法进行粗匹配, 为加速特征匹配环节, 舍弃 GMS 原算法的扩展运动核, 忽略旋转变换和尺度变换对匹配精度的影响, 快速筛选并剔除误匹配。再丢弃临近图像边缘和匹配中心 3×3 邻域内存在明显像素差的匹配以优化匹配点候选序列。

设输入两帧图像为$\{I_a, I_b\}$, $C=\{c_1, c_2, \cdots, c_i, \cdots, c_n\}$为图像$I_a$与$I_b$之间所有最近邻特征匹配的集合, 其中, c_i表示特征点a_i和b_i的匹配, 定义c_i的邻域N_i和相似领域S_i分别为

$$N_i=\left\{c_j \mid c_j\in C, c_j\neq c_i, d(a_i,a_j)<r_1\right\} \tag{6}$$

$$S_i=\left\{c_j \mid c_j\in N_i, d(b_i,b_j)<r_2\right\} \tag{7}$$

式中: $d(\cdot,\cdot)$为两点间欧氏距离; r_1和r_2为距离阈值; 定义c_i的运动支持度$|S_i|$, 表示S_i中的匹配点数量。

在处理动态场景时, 如果不存在动态物体或动态物体在连续图像帧间没有明显移动, 则匹配点邻域内运动支持度较高, 相反, 当受到动态物体明显运动的影响时, 邻域内特征匹配误差较大, 使得其支持度较低。因此, 统计邻域内运动支持度$|S_i|$就可以有效区分动态场景下的正误匹配。假设匹配点间相互独立, 则$|S_i|$的概率分布模型近似满足二项分布:

$$|S_i|\sim\begin{cases}B(|N_i|,p_{\mathrm{t}}) & c_i\in T\\ B(|N_i|,p_{\mathrm{f}}) & c_i\in F\end{cases} \tag{8}$$

式中: $B(\cdot,\cdot)$表示二项分布; $|N_i|$为邻域N_i内匹配点数量; p_{t}和p_{f}分别为该邻域内正确匹配和错误匹配的运动支持概率; T和F分别为正确匹配和错误匹配的集合, 该二项分布模型的期望和标准差为

$$E_{|S_i|}=\begin{cases}E_{\mathrm{t}}=|N_i|p_{\mathrm{t}} & c_i\in T\\ E_{\mathrm{f}}=|N_i|p_{\mathrm{f}} & c_i\in F\end{cases} \tag{9}$$

$$S_{|S_i|}=\begin{cases}S_{\mathrm{t}}=\sqrt{|N_i|p_{\mathrm{t}}(1-p_{\mathrm{t}})} & c_i\in T\\ S_{\mathrm{f}}=\sqrt{|N_i|p_{\mathrm{f}}(1-p_{\mathrm{f}})} & c_i\in F\end{cases} \tag{10}$$

实际情况下, 正确匹配数量远大于错误匹配, 根据$|S_i|$的期望和标准差, 计算二者之间的区分度P_i:

$$P_i=\frac{E_{\mathrm{t}}-E_{\mathrm{f}}}{S_{\mathrm{t}}+S_{\mathrm{f}}}=\frac{|N_i|(p_{\mathrm{t}}-p_{\mathrm{f}})}{\sqrt{|N_i|p_{\mathrm{t}}(1-p_{\mathrm{t}})}+\sqrt{|N_i|p_{\mathrm{f}}(1-p_{\mathrm{f}})}} \tag{11}$$

由式 (6) 可得出, $P_i\propto\sqrt{|N_i|}$, 即正误匹配区分度P_i同$\sqrt{|N_i|}$正相关, 说明同一邻域内特征匹配数目越多, 正误匹配间的可区分度就越强, 利用这一特性, 设定阈值τ_i:

$$\tau_i=\alpha\sqrt{|N_i|} \tag{12}$$

式中: α为超参数, 通常取 4～6。如果某一邻域内的运动支持度$|S_i|$大于阈值τ_i, 就认为特征点a_i和b_i是一对正确匹配, 否则认为可能存在大量动态物体的干扰, 删除误匹配。这样既可以消除局部动态区域的不匹配, 又可以提高静态区域的整体匹配质量。

1.2.2　动态场景划分

若场景中存在大量动态信息的干扰, 经 GMS 算法筛选保留的特征匹配点总数和检测框内匹配点数量会明显减少, 匹配点分布均匀程度下降。根据得到的特征匹配筛选结果, 结合目标检测获得的语义先验信息和以下 3 个指标计算当前帧的特征匹配精度, 作为区分场景动态程度的依据。

1) 特征匹配点数量。充足的匹配点可以提供丰富的几何和纹理特征, 有助于提高匹配的准确性和稳定性。因此, 每帧图像获得的匹配点总数应至少为提取特征点总数n的 30%, 并记录当前帧匹配点总数为n_{t}。否则, 认为匹配精度不佳, 先前 GMS 算法滤除了大量误匹配, 环境中可能存在大量的运动干扰, 直接将当前帧划分至高动态类别。

2) 检测框内匹配点数量。实际情况下, 检测框内存在动态点的概率远大于框外背景点。如若受到运动目标的影响, 经 GMS 算法筛选过滤后, 检测

框内会缺少有效的匹配点，得到的少量剩余匹配点也并不可靠。因此，如式 (13) 和式 (14) 所示，根据比例阈值筛选出框内匹配点数量不足的潜在动态框，并对潜在动态框内的匹配点进行计数：

$$f(n_i^{\mathrm{b}}, s_i^{\mathrm{b}}) = \begin{cases} 1 & n_i^{\mathrm{b}}/n_{\mathrm{t}} < \mathrm{s}_i^{\mathrm{b}}/2s_{\mathrm{t}} \\ 0 & \text{其他} \end{cases} \tag{13}$$

$$n_{\mathrm{t}}^{\mathrm{b}} = \sum_{i=1}^{n} f(n_i^{\mathrm{b}}, n_i^{\mathrm{b}}) n_i^{\mathrm{b}} \tag{14}$$

式中：n_i^{b}为第i个检测框内匹配点的数量；s_i^{b}和s_{t}分别为第i个检测框的面积和当前帧图像的总面积。理想状态下，检测框内匹配点数目与检测框面积大小成正比，设置阈值为s_i^{b}与s_{t}的比值，如果n_i^{b}与n_{t}的比值小于阈值的1/2，则认为检测框内可能存在大量已被剔除的动态点，将当前检测框标记为潜在动态属性，记录全部此类检测框内匹配点总数$n_{\mathrm{t}}^{\mathrm{b}}$。

3）匹配点分布均匀程度：均匀分布的匹配点可以避免在相同区域内冗余匹配。同时，当某些区域由于运动扰动而无法提供有效匹配时，可以确保其他区域仍有足够的匹配信息，从而提高鲁棒性。此处采用四叉树策略评估匹配点分布均匀程度，将图像划分成大小均匀的4个子区域，式 (15) 和式 (16) 用于根据比例阈值筛选出匹配点过度聚集和分散的异常子区域，并对异常子区域内匹配点进行计数：

$$f(n_i^{\mathrm{r}}, s_i^{\mathrm{r}}) = \begin{cases} 1 & \dfrac{n_i^{\mathrm{r}}}{n_{\mathrm{t}}} > \dfrac{2s_i^{\mathrm{r}}}{s_{\mathrm{t}}} \cup \dfrac{n_i^{\mathrm{r}}}{n_{\mathrm{t}}} < \dfrac{s_i^{\mathrm{r}}}{2s_{\mathrm{t}}} \\ 0 & \text{其他} \end{cases} \tag{15}$$

$$n_{\mathrm{t}}^{\mathrm{r}} = \sum_{i=1}^{n} f(n_i^{\mathrm{r}}, s_i^{\mathrm{r}}) n_i^{\mathrm{r}} \tag{16}$$

式中：n_i^{r}为第i个区域内匹配点的数量；s_i^{r}为第i个区域的面积。理想状态下，划分区域内匹配点数目与该区域图像面积大小成正比，同样将阈值设置为s_i^{r}与s_{t}的比值。

图4中，红色点表示当前帧的匹配点，如果n_i^{r}与n_{t}的比值大于阈值的2倍，则将这一区域判定为聚集区域，图中用绿色对角线标记；如果n_i^{r}与n_{t}的比值小于阈值的1/2，则将这一区域判定为分散区域，图中用蓝色对角线标记。受运动干扰造成的匹配点分布过度聚集和发散会显著影响整体匹配效果，故将上述两类区域内匹配点数目计入$n_{\mathrm{t}}^{\mathrm{r}}$，如已计入$n_{\mathrm{t}}^{\mathrm{b}}$则不再重复统计。反之，则对该区域进行下一次划分，直至理想状态下区域内匹配点数目小于2不再可分。

除去上述异常区域内的匹配点，最后剩余的有效匹配点数目n_{r}为

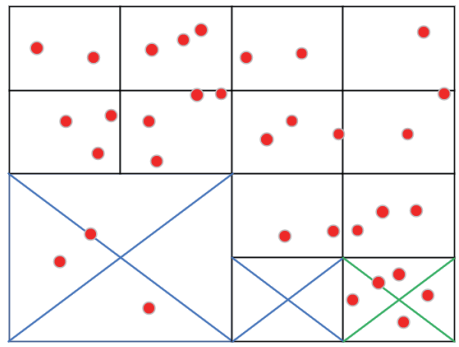


图4 四叉树策略划分图像区域

Fig. 4 Quadtree strategy for dividing image regions

$$n_{\mathrm{r}} = n_{\mathrm{t}} - n_{\mathrm{t}}^{\mathrm{b}} - n_{\mathrm{t}}^{\mathrm{r}} \tag{17}$$

定义当前帧的匹配精确度P，如下：

$$P = n_{\mathrm{r}}/n_{\mathrm{t}} \tag{18}$$

如果$P \geqslant 0.5$，则认为当前帧取得理想的特征匹配效果，场景中只存在轻微的运动干扰，标记当前帧为低动态帧，可以直接将GMS算法得到的匹配结果用于SLAM系统的地图初始化，最小化3D地图点和2D特征点的重投影误差：

$$\boldsymbol{T}_{\mathrm{cw}} = \arg\min_{\boldsymbol{T}} \frac{1}{2} \sum_{i=1}^{n} \| \boldsymbol{x}_i - \pi(\boldsymbol{T}_{\mathrm{cw}} \boldsymbol{p}_i) \|_2^2 \tag{19}$$

式中：

$$\pi\left(\begin{bmatrix} X \\ Y \\ Z \end{bmatrix}\right) = \begin{bmatrix} f_x X/Z + c_x \\ f_y Y/Z + c_y \end{bmatrix} \tag{20}$$

其中：$\boldsymbol{T}_{\mathrm{cw}}$为相机位姿，其李群表示为$\boldsymbol{T}$；$\boldsymbol{x}_i$为2D静态特征点；$\boldsymbol{p}_i$为与$\boldsymbol{x}_i$对应的3D地图点；$\pi(\cdot)$为相机的投影方程；$f_x$和$f_y$为$x$和$y$方向的相机焦距；$(c_x, c_y)$为相机光心坐标。

通过使用李代数，构建无约束的优化问题，利用高斯牛顿法进行迭代求得最优解，可获得准确的相机位姿。

若$P < 0.5$，则认为动态物体对特征匹配环节产生严重干扰，标记当前帧为高动态帧，后续将执行进一步检查。

1.3 引入卡方检验的运动一致性检测

针对筛选得到的高动态帧，采用与低动态帧不同的处理策略：先计算光流金字塔，获取当前帧中匹配的特征点，再利用对极几何约束检测物体与相机运动的一致性。

运动一致性检测的误差项是求解基础矩阵得到的匹配点到对应极线的距离，如果极线距离超过一定范围ε，就认为该点属于动态离群点。

然而，确定合理的约束条件存在难度。如果条件宽松，就不能有效检测出全部动态特征点；而条件严格，就会使原本可以保留的静态特征点被错误

删除。当前的主流方法在设定阈值时通常取 1 像素，但这种固定阈值缺乏灵活性，难以适应复杂多变的现实场景。为有效解决上述问题，本文算法选取当前帧中检测框外的背景点和上一帧的静态点作为样本，组成正态分布模型，并在此基础上引入卡方检验识别动态特征点，确保算法在实际应用中保持准确、高效，同时提升其对不同环境的适应性。

如图 5 所示，假设 O_1 和 O_2 是相机的光心，空间点 X 在图像帧 I_1 中有一个特征点 $\boldsymbol{x}_1$ 与之对应，并与当前帧 I_2 中的特征点 $\boldsymbol{x}_2$ 匹配，其的归一化坐标分别为

$$\begin{cases}\boldsymbol{x}_1=[u_1,v_1,1]\\ \boldsymbol{x}_2=[u_2,v_2,1]\end{cases} \tag{21}$$

式中：u、v 为像素平面的坐标。计算得到当前帧的极线 $\boldsymbol{l}_2$ 为

$$\boldsymbol{l}_2=\boldsymbol{F}\boldsymbol{x}_1=\boldsymbol{F}[u_1,v_1,1]=[\boldsymbol{a},\boldsymbol{b},\boldsymbol{c}] \tag{22}$$

其中：$\boldsymbol{F}$ 为基础矩阵；$\boldsymbol{a}$、$\boldsymbol{b}$、$\boldsymbol{c}$ 为列向量。理想状态下对极几何约束的表达式为

$$\boldsymbol{x}_2^{\mathrm{T}}\boldsymbol{F}\boldsymbol{x}_1=\boldsymbol{x}_2^{\mathrm{T}}\boldsymbol{l}_2=\boldsymbol{0} \tag{23}$$

图中空间点 X 运动至 X'，$\boldsymbol{x}_2'=[u_2',v_2',1]$ 到极线 $\boldsymbol{l}_2$ 距离 d 为

$$d=\frac{|au'_2+bv'_2+c|}{\sqrt{a^2+b^2}}>\varepsilon \tag{24}$$

当前待测点极线距离 d_{c} 和静态样本点极线距离 d_i^{s} 之间的误差项为

$$e_i=\left(d_{\mathrm{c}}-\frac{1}{n}\sum_{i=1}^{n}d_i^{\mathrm{s}}\right)/\sigma \tag{25}$$

式中：n 为样本中的静态点个数；σ 为归一化平面的标准差；误差 e_i 服从一维标准正态分布，根据卡方分布：

$$\mathrm{chsq}(x;k)=\begin{cases}\dfrac{x^{(k/2-1)}\mathrm{e}^{-x/2}}{2^{k/2}\Gamma(k/2)} & x>0\\ 0 & x\leqslant 0\end{cases} \tag{26}$$

式中：k 为卡方分布的自由度；函数 $\Gamma(v)$ 的定义为

$$\Gamma(v)=\int_0^{\infty}\mathrm{e}^{-t}t^{v-1}\mathrm{d}t\quad v>0 \tag{27}$$

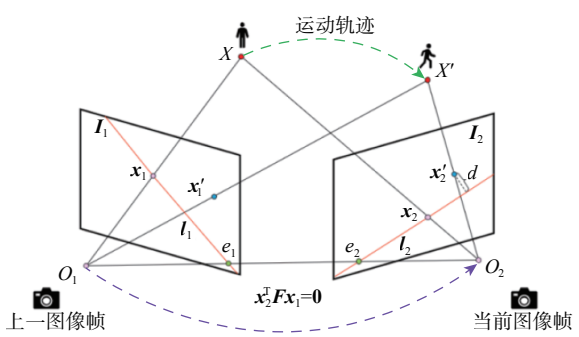


图 5　对极几何约束

Fig. 5　Epipolar geometry constraints

对匹配点对进行对称投影操作，得到 2 组对称极线距离，再计算单次估计结果：

$$\Delta_{\mathrm{d}}=\mathrm{chsq}\left((e_i)^2;2\right) \tag{28}$$

设置 $d_{\mathrm{th}}=\chi^2_{0.05}(2)$，即自由度为 2 的卡方分布在显著性水平为 0.05 时的临界阈值，超过临界阈值的特征点被认为是动态点，此过程计算复杂度较小，且得到的动态特征点相比原方法具有更高的可靠性。

1.4　动态特征剔除策略

经运动一致性检测得到动态特征点，再利用先前获取的语义信息对检测框内的特征点进行进一步处理。为避免直接删除检测框内全部特征点导致 SLAM 系统缺少关键数据而失效，提出一种动态特征剔除策略，最大限度地保留活跃的静态点用于相机位姿初始化。根据检测框内动态特征点的比例设置自适应阈值，可以有效识别不稳定的数据关联，提高系统在处理动态目标干扰时的鲁棒性。

本文算法考虑所有被检测到的目标都可能处于运动状态，如图 6 所示，第 1 行表示由语义模块检测到的检测框；第 2 行是运动一致性检测的结果，动态点被标记为蓝色，静态点被标记为绿色；第 3 行展示了结合语义信息与几何约束的动态特征剔除效果。

算法 1　动态特征剔除。

输入：人类检测框 B_n^{H}，其他类检测框 B_n^{O}，当前帧特征点集合 P_n，当前帧动态特征点集合 P_n^{d}。

输出：静态特征点集合 S_n。

1. $S_n\leftarrow P_n$
2. for each $B_n^{\mathrm{H}}(i)$ in B_n^{H} do
3. $s_h\leftarrow$特征点 P_n 总数 in $B_n^{\mathrm{H}}(i)$
4. $o_h\leftarrow$动态特征点 P_n^{d} 总数 in $B_n^{\mathrm{H}}(i)$
5. if $s_h>0.25o_h$ then
6. $S_n\leftarrow S_n\backslash B_n^{\mathrm{H}}(i)$
7. else
8. for each 特征点 (u_n,v_n) in $B_n^{\mathrm{H}}(i)$ do
9. for each 动态特征点 (du_n,dv_n) in $B_n^{\mathrm{H}}(i)$ do
10. if (u_n,v_n) in $B_n^{\mathrm{H}}(i)$ and (du_n,dv_n) in $B_n^{\mathrm{H}}(i)$ and $\sqrt{(u_n-du_n)^2+(v_n-dv_n)^2}\leqslant 15$ then
11. $S_n\leftarrow S_n\backslash(u_n,v_n)$
12. end if
13. end for
14. end for
15. end if
16. end for
17. for each $B_n^{\mathrm{O}}(i)$ in B_n^{O} do
18. $s\leftarrow$特征点 P_n 总数 in $B_n^{\mathrm{O}}(i)$



(a) 图像帧1 (b) 图像帧2 (c) 图像帧3 (d) 图像帧4 (e) 图像帧5 (f) 图像帧6 (g) 图像帧7

图 6 动态特征剔除结果

Fig. 6 Results of dynamic feature culling

19. $o \leftarrow$动态特征点P_n^{d}总数 in $B_n^{\mathrm{O}}(i)$

20. if $s > 0.3o$ then

21. $S_n \leftarrow S_n \backslash B_n^{\mathrm{O}}(i)$

22. end if

23. end for

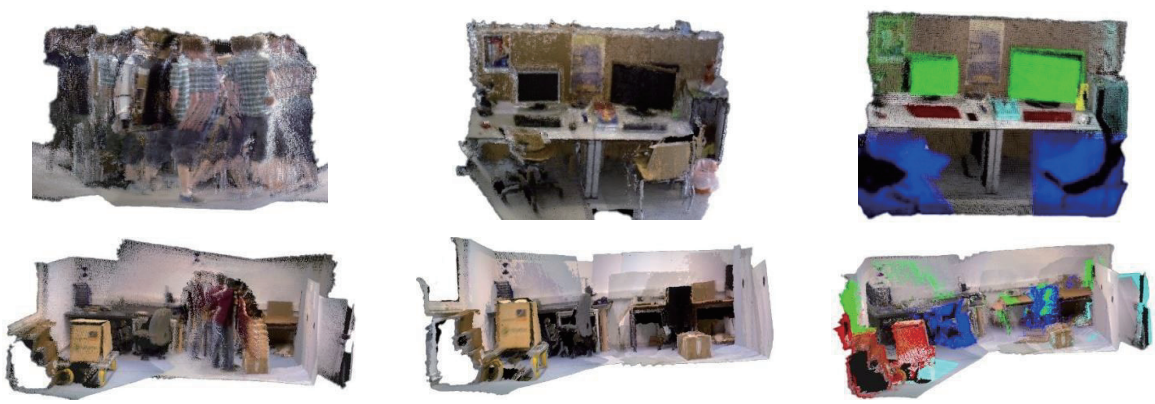
动态特征剔除流程如算法 1 所示，如果检测框内的动态特征点超过框内特征点总数的 30%，则标记检测框为动态，删除框内所有特征点。如图 6(a)～图 6(c) 所示，视角内动态人框内特征点被有效检测和移除。对于人类这类可以进行局部运动的非刚性物体，删除动态点和以其为中心，半径为 15 像素的圆域内的其他特征点，保留其余的静态点，若最初的动态点比例超过 25% 就舍弃整个检测框。本文算法只移除同时存在于静态框外和动态框内的特征点，从图 6(b)～图 6(e) 可以看出，当动态人框和静态显示器框出现重叠时，在保留显示器框内的特征点的基础上，再移除其余的动态点。此外，本文算法能够识别图 6(f)～图 6(g) 中视角内出现的不完整目标及其语义类别并准确去除动态离群点。

此策略不需要预先定义对象类的运动属性，对于不同的对象类具有一致的鲁棒性。设置百分比阈值允许算法根据不同的检测框自动调整可容忍的动态异常点个数，从而有效应对遮挡，提升动态框判别的准确性。

1.5 稠密重建

为获取详细的环境特征信息，帮助机器人更好地理解环境，本文算法在 ORB-SLAM3 基础上增设用于稠密重建的并行线程。与跟踪线程不同，在地图构建环节，移除关键帧中所有潜在动态目标检测框内的像素区域，再对局部点云进行变换和合并，以获得只包含场景静态信息的全局点云地图。此外，利用 YOLOX-s 获取关键帧中目标对象的语义标签和检测框位置信息，并将其投影至对应的点云空间，赋予三维点云二维图像的语义信息。通过附加的语义信息，机器人能够精确辨识环境中物体的类别，有助于其执行更复杂的任务规划和决策。

在 TUM 和 BONN 数据集中各选择一个图像序列验证地图重建效果如图 7 所示。如图 7(a) 所示，ORB-SLAM3获得的稠密地图中，移动的人产生大量的重影，覆盖大量其他场景信息，而本文算法能



(a) 初始点云 (b) 静态点云 (c) 语义点云

图 7 稠密点云重建结果

Fig. 7 Results of dense point cloud reconstruction

够获得清晰的静态点云 (见图 7(b)), 生成的语义地图 (见图 7(c)) 对场景中的静态物体进行标注, 可用于机器人后续执行物体识别、路径规划等高级任务。

2　本文算法

本文算法步骤如下:

步骤 1　初始化。选择相机模式, 输入 RGB 图像和深度图像。

步骤 2　目标检测获取语义先验, 并对目标检测缺失进行补偿。

1) 目标检测。将 RGB 图像输入 YOLOX-s 目标检测网络, 输出包含检测框的置信度、语义类别及坐标信息的当前帧中目标检测结果 $\boldsymbol{D}$。

2) 轨迹预测。输入 $\boldsymbol{D}$ 中潜在运动目标的检测结果 $\boldsymbol{D}_t$ 进行目标跟踪, 根据式 (1) 利用卡尔曼滤波器进行轨迹预测, 获得预测结果 $\boldsymbol{T}_{\mathrm{p}}$。

3) 检测缺失判定。根据式 (3) 计算检测框 D_i 与预测框 P_i 匹配相似度, 判断相似度是否小于设定阈值, 若是, 即认为出现检测缺失, 使用预测框作为当前帧的目标检测结果进行检测缺失补偿。

4) 数据关联。建立代价矩阵 $\boldsymbol{I}_{\mathrm{oU}}(\boldsymbol{D},\boldsymbol{P})$, 使用匈牙利算法进行 2 次数据关联, 使得当前帧总体匹配相似度最大, 输出有效匹配轨迹 $\boldsymbol{T}_{\mathrm{m}}^{1}$ 和 $\boldsymbol{T}_{\mathrm{m}}^{2}$。

5) 轨迹更新。输入有效匹配轨迹 $\boldsymbol{T}_{\mathrm{m}}^{1}$ 和 $\boldsymbol{T}_{\mathrm{m}}^{2}$, 根据式 (5) 利用卡尔曼滤波器进行轨迹更新, 输出全部跟踪轨迹 $\boldsymbol{T}_t$。

步骤 3　特征匹配筛选。依据式 (6)～式 (12) 由 GMS 算法快速筛选过滤粗匹配结果 $\boldsymbol{C}$, 获得有效匹配 $\boldsymbol{C}_{\mathrm{t}}$。

步骤 4　动态场景划分。结合步骤 2 获得的目标检测结果 $\boldsymbol{D}$ 与步骤 3 获得的特征匹配筛选结果 $\boldsymbol{C}_{\mathrm{t}}$ 计算特征匹配精度 P_r, 根据特征匹配精度 P_r 将图像帧划分为低动态帧和高动态帧。

1) 判断是否满足 $n_{\mathrm{t}} < 0.3n$, 若是, 则划分当前帧为高动态帧; 否则转至步骤 2)。

2) 判断是否满足 $\dfrac{n_i^{\mathrm{b}}}{n_{\mathrm{t}}} < \dfrac{s_i^{\mathrm{b}}}{2s_{\mathrm{t}}}$, 若是, 则对满足条件的 n_i^{b} 计数为 $n_{\mathrm{t}}^{\mathrm{b}}$, 转至步骤 3); 否则, 不进行计数, 转至步骤 3)。

3) 判断是否满足 $\dfrac{n_i^{\mathrm{r}}}{n_{\mathrm{t}}} > \dfrac{2s_i^{\mathrm{r}}}{s_{\mathrm{t}}} \cup \dfrac{n_i^{\mathrm{r}}}{n_{\mathrm{t}}} < \dfrac{s_i^{\mathrm{r}}}{2s_{\mathrm{t}}}$, 若是, 则对满足条件的 n_i^{r} 计数为 $n_{\mathrm{t}}^{\mathrm{r}}$, 转至步骤 4); 否则, 不进行计数, 转至步骤 4)。

4) 依据式 (17) 和式 (18), 获得匹配精度 P_r, 判断是否满足 $P_r \geqslant 0.5$, 若是, 则划分当前帧为低动态帧, 将有效匹配点 C_{t} 作为当前帧静态点集合 S_n; 否则, 划分当前帧为高动态帧。

步骤 5　运动一致性检测。将步骤 4 获得的高动态帧输入 LK 光流, 获得光流特征点, 依据式 (21)～式 (28) 利用引入卡方分布的对极几何约束判断光流特征点是否与相机运动具有一致性, 若是, 则判定该点为静态点; 否则, 判定该点为动态点。

步骤 6　动态特征剔除。对于高动态帧, 依据算法 1 输入步骤 2 获得检测框 B_n 和步骤 5 获得的动态点集合 P_n^{d}, 输出当前帧静态点集合 S_n。

步骤 7　稠密重建。输入步骤 4 和步骤 6 得到的静态点集合 S_n, 经跟踪线程、局部建图线程和稠密重建线程处理, 生成具有全局一致性的稠密点云地图。

3　实验测试

为全面评估本文 SLAM 算法的综合性能, 使用绝对轨迹误差 (absolute trajectory error, ATE) 和相对位姿误差 (relative pose error, RPE) 作为参考指标, 其的均方根误差 (root mean squared error, RMSE) 和标准差 (standard deviation, S.D.) 反映算法的精度和可靠性。

首先, 在广泛使用的 3 个公开数据集上测试跟踪精度; 随后, 进行消融实验, 验证基线算法外其他各单一模块的有效性; 最后, 进行运行时长分析, 并在包含动态对象的现实场景中评估所提出算法的实时性能。

本文算法的实验平台是一台运行环境为 Ubuntu 20.04 的笔记本电脑, CPU 型号为 Intel i9, GPU 型号为 Nvidia Geforce RTX 4060 Laptop, 运行内存 RAM 为 32 GB。

3.1　室内场景测试

为充分检验本文 SLAM 算法在室内动态场景中的表现, 在本项实验中, 使用 TUM RGB-D 数据集[19] 对比本文算法与 ORB-SLAM3[14] 及其他集成目标检测网络的先进动态 SLAM 算法, 包括 Crowd-SLAM[11]、SG-SLAM[12] 和 OVD-SLAM[13]。

TUM RGB-D 数据集由慕尼黑工业大学计算机视觉小组在不同的室内场景中使用 Microsoft Kinect 传感器记录的 39 个序列组成, 同时包含轨迹真值和相机参数信息。实验选用其中针对动态场景任务的 4 个高动态序列和 4 个低动态序列进行实验分析。其中, “walking” 属于高动态序列, 展示了 2 个人在办公桌周围行走的场景, 旨在评估视觉 SLAM 算法处理在场景中大量快速移动的动态对象的能力。“sitting” 属于低动态序列, 描述了 2 个人坐在办公桌前交谈的情形, 旨在评估视觉 SLAM 算法处理缓慢移动的动态对象的能力。“half, rpy,

static, xyz”则表示相机沿不同方向的运动。

为避免不同的实验设备对结果产生额外影响，所有实验数据均源自本文所搭建的实验平台，利用每个算法ATE的RMSE和S.D.以评估其全局一致性和稳定性，并利用平移和旋转RPE的RMSE和S.D.评估其局部定位精度和鲁棒性。

如表1～表3所示，本文算法在“sitting”低动态序列中平均定位精度最高，由于设计了区分场景动态程度的策略，更多的图像帧被分类为低动态场景，因此保留了更多有利于跟踪的特征点，仅在fr3/s/xyz中精度略低于ORB-SLAM3。在“walking”高动态序列中，动态对象的运动范围和幅度较大，本文算法有效地剔除了大量动态特征点，ATE和RPE均较ORB-SLAM3有明显下降，平均提升超过90%。本文算法在fr3/w/rpy序列的ATE和RPE高于OVD-SLAM，主要原因可能是相机视角转换较大造成特征匹配精度下降，同时场景中人的运动方向与极线方向相似，导致算法有不同程度的退化。但总体而言，本文算法的综合实验结果优于其他算法，所有序列的平均精度达到了最先进的水平。

表1 TUM数据集绝对轨迹误差结果

Table 1 Results of ATE on TUM dataset m

序列	RMSE					S.D.				
	ORB-SLAM3[14]	ORB-SLAM3[11]	ORB-SLAM3[12]	ORB-SLAM3[13]	本文算法	ORB-SLAM3[14]	ORB-SLAM3[11]	ORB-SLAM3[12]	ORB-SLAM3[13]	本文算法
fr3/s/half	0.032 5			0.022 6	**0.019 6**	0.012 9			0.011 7	**0.011 0**
fr3/s/rpy	0.022 8			0.035 5	**0.020 6**	0.014 6			0.025 9	**0.009 5**
fr3/s/static	0.007 9	0.008 5	0.008 0	0.006 5	**0.006 3**	0.003 9	0.004 6	0.004 0	0.002 9	**0.002 8**
fr3/s/xyz	**0.009 4**	0.016 4	0.012 9	0.012 8	0.011 6	**0.004 6**	0.008 9	0.006 7	0.008 8	0.007 2
fr3/w/half	0.191 5	0.023 7	0.022 7	0.019 2	0.018 9	0.069 2	0.011 3	0.012 6	**0.009 1**	0.009 9
fr3/w/rpy	0.562 7	0.060 9	0.046 0	**0.032 4**	**0.034 2**	0.211 0	0.042 1	0.030 1	**0.018 0**	0.019 0
fr3/w/static	0.027 8	0.007 4	0.007 6	0.006 6	**0.006 3**	0.022 1	0.003 5	0.004 1	0.003 1	**0.002 9**
fr3/w/xyz	0.395 7	0.017 3	0.015 0	0.014 5	**0.013 5**	0.151 8	0.009 6	0.008 3	**0.006 9**	**0.006 9**

注：每个序列的最优结果用粗体突出显示，次优结果用下划线突出显示。

表2 TUM数据集平移漂移结果

Table 2 Results of translational drifton TUM dataset m

序列	RMSE					S.D.				
	ORB-SLAM3[14]	ORB-SLAM3[11]	ORB-SLAM3[12]	ORB-SLAM3[13]	本文算法	ORB-SLAM3[14]	ORB-SLAM3[11]	ORB-SLAM3[12]	ORB-SLAM3[13]	本文算法
fr3/s/half	**0.007 8**			0.016 6	0.014 1	**0.004 2**			0.009 6	0.008 7
fr3/s/rpy	0.013 6			0.019 1	**0.009 7**	0.009 2			0.013 8	**0.005 5**
fr3/s/static	0.005 8	0.006 0	0.009 0	0.005 7	**0.004 9**	0.003 1	0.003 1	0.004 7	0.002 9	**0.002 5**
fr3/s/xyz	**0.008 4**	0.014 1	0.011 9	0.012 0	0.009 6	**0.004 3**	0.008 5	0.006 3	0.007 9	0.005 5
fr3/w/half	0.337 3	0.019 3	0.018 0	**0.013 8**	**0.013 8**	0.190 8	0.011 5	0.011 1	**0.007 4**	0.008 0
fr3/w/rpy	0.863 2	0.042 2	0.030 3	**0.020 1**	0.021 1	0.457 0	0.030 5	0.023 5	**0.012 0**	0.012 5
fr3/w/static	0.040 5	0.007 4	0.007 7	**0.005 6**	0.006 0	0.031 0	0.003 8	0.004 3	**0.002 7**	0.003 2
fr3/w/xyz	0.607 5	0.015 2	0.013 8	0.013 0	**0.012 2**	0.370 4	0.009 5	0.008 1	0.007 1	**0.006 7**

注：每个序列的最优结果用粗体突出显示，次优结果用下划线突出显示。

表3 TUM数据集旋转漂移结果

Table 3 Results of rotational drifton TUM dataset m

序列	RMSE					S.D.				
	ORB-SLAM3[14]	ORB-SLAM3[11]	ORB-SLAM3[12]	ORB-SLAM3[13]	本文算法	ORB-SLAM3[14]	ORB-SLAM3[11]	ORB-SLAM3[12]	ORB-SLAM3[13]	本文算法
fr3/s/half	**0.981 6**			1.034 6	1.001 4	**0.446 8**			0.493 4	0.499 7
fr3/s/rpy	0.955 2			1.046 8	**0.894 7**	0.451 1			0.533 9	**0.438 5**
fr3/s/static	0.384 6	0.351 8	0.327 7	0.335 3	**0.313 1**	0.167 5	0.161 9	**0.149 7**	0.157 6	0.153 3
fr3/s/xyz	0.606 8	0.694 1	0.628 2	0.668 1	**0.600 9**	0.311 5	0.322 6	0.320 1	0.349 8	**0.308 8**
fr3/w/half	6.930 9	1.223 4	1.019 0	0.926 2	**0.880 8**	4.138 8	0.652 8	0.540 3	**0.425 8**	0.455 7
fr3/w/rpy	15.564 7	2.077 1	1.342 6	**0.984 2**	1.066 3	9.287 3	1.246 7	0.748 2	**0.504 9**	0.564 9
fr3/w/static	0.658 0	0.312 2	0.308 6	0.279 8	**0.265 7**	0.431 1	0.138 0	0.139 1	**0.121 7**	0.122 3
fr3/w/xyz	9.083 1	0.659 0	0.663 2	0.631 8	**0.615 4**	6.001 5	0.333 1	0.348 5	0.325 6	**0.321 9**

注：每个序列的最优结果用粗体突出显示，次优结果用下划线突出显示。

图 8 展示了本文算法与 ORB-SLAM3 的轨迹估计结果和轨迹真值的对比。相较于 ORB-SLAM3, 在图 8(a)～图 8(d) 所示的低动态场景中, 本文算法跟踪精度有一定程度的提升。更为显著的是, 图 8(e)～图 8(h) 中可以直观地看出, 本文算法大幅降低了高动态场景下 ORB-SLAM3 的轨迹跟踪误差。

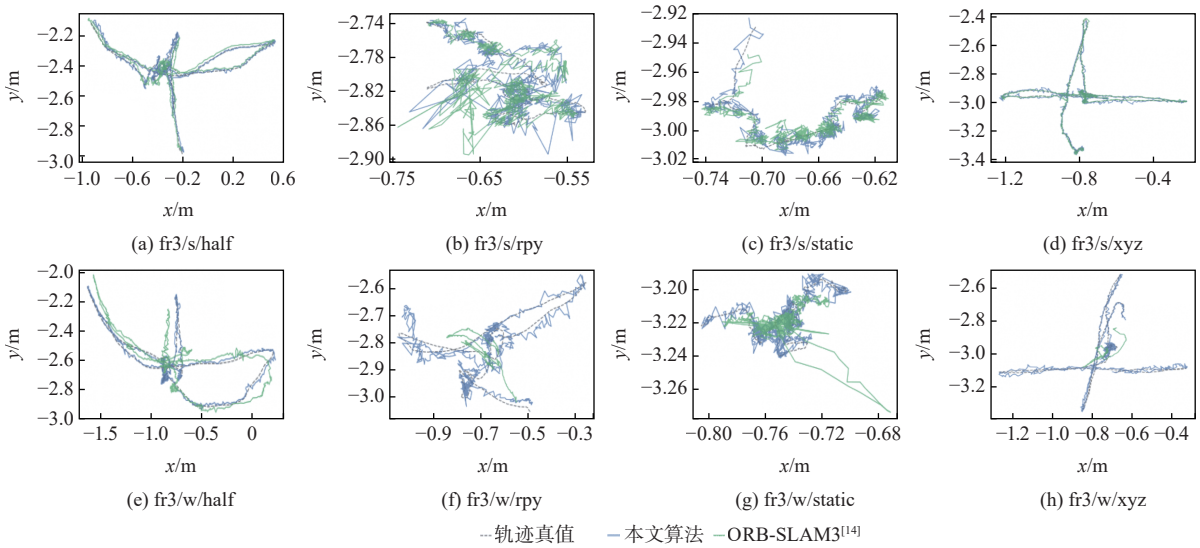


图 8　TUM 数据集轨迹对比

Fig. 8　Comparison of trajectories on TUM dataset

为进一步评估本文算法在包含大量动态物体的室内复杂环境中的性能, 使用 BONN RGB-D 数据集[20] 将本文算法同 ORB-SLAM3、Crowd-SLAM、SG-SLAM 和 OVD-SLAM 进行对比。

BONN RGB-D 数据集由波恩大学在 2019 年发布, 使用 ASUS Xtion Pro LIVE 相机获取图像, Optitrack Prime 13 运动捕捉系统记录轨迹真值, 其中包含 24 个高动态序列, 序列格式与 TUM RGB-D 数据集相同。与 TUM 数据集相比, BONN 数据集中的动态场景更具挑战性。选择其中具有代表性的 7 个序列进行测试, 其中“crowd”序列是 5 个人在房间里随机行走的场景。“moving no box”序列展示了 1 个人把 1 个盒子从地面搬到桌子上的过程。“person tracking”序列是摄像机追踪 1 个行走的人的场景。

表 4 记录了上述 SLAM 算法的 ATE, 结果表明, 本文算法在大多数测试序列中优于同类算法, 特别是在存在高密度动态物体的 3 个“crowd”序列中, 与 ORB-SLAM3 相比, 定位精度提升幅度超过 95% 展现出超过 95% 的显著提升。这不仅再次证明了本文算法在动态场景下达到了先进的精度, 还验证了其通用性。如图 9 所示, 在图 9(a)～图 9(c) 表示的“crowd”序列中, ORB-SLAM3 受到多个行走的人的干扰, 导致大量图像帧跟踪丢失; 在图 9(f)～图 9(g) 表示“person tracking”序列中, 相机视角跟随行走的人进行转换, 导致其位姿产生了连续的错误估计, 从而使得估计轨迹与真值相比存在明显误差和漂移。相比之下, 本文算法能够获得比 ORB-SLAM3更为准确的估计轨迹, 进一步证明了本文算法出色的场景泛化能力。

表 4　BONN 数据集绝对轨迹误差结果

Table 4　Results of ATE on BONN dataset　　m

序列	RMSE					S.D.				
	ORB-SLAM3[14]	ORB-SLAM3[11]	ORB-SLAM3[12]	ORB-SLAM3[13]	本文算法	ORB-SLAM3[14]	ORB-SLAM3[11]	ORB-SLAM3[12]	ORB-SLAM3[13]	本文算法
crowd	0.331 7	0.017 0	0.018 7	0.016 9	**0.015 3**	0.211 3	0.008 6	0.008 6	0.008 0	**0.006 5**
crowd2	1.488 5	0.033 3	0.032 2	0.025 9	**0.019 4**	0.595 8	0.019 9	0.021 1	0.019 2	**0.009 4**
crowd3	0.459 3	0.039 8	0.028 9	0.021 8	**0.020 2**	0.221 9	0.022 0	0.017 9	0.012 0	**0.010 5**
move_no_box	0.325 6	**0.012 8**	0.017 7	0.013 7	0.013 8	0.154 0	**0.006 0**	0.012 5	0.006 8	0.006 4
move_no_box2	0.038 2	0.024 0	**0.019 0**	0.023 0	0.021 5	0.016 9	0.011 1	0.008 5	0.008 5	**0.008 2**
person_tracking	0.561 1	0.034 3	0.031 1	0.036 0	**0.026 4**	0.229 9	0.020 4	0.014 9	0.014 9	**0.013 0**
person_tracking2	0.753 8	0.034 6	0.034 3	0.034 9	**0.032 0**	0.416 9	**0.011 7**	0.012 2	0.013 0	0.012 4

注: 每个序列的最优结果用粗体突出显示, 次优结果用下划线突出显示。

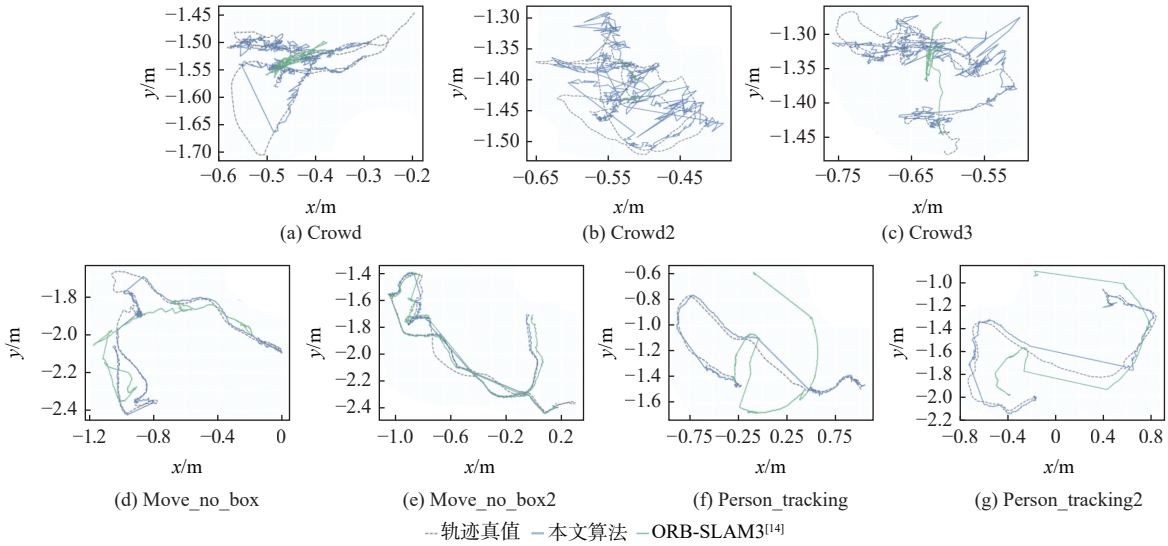


图 9 BONN 数据集轨迹对比

Fig. 9 Comparison of trajectories on BONN dataset

3.2 室外场景测试

为充分检验本文 SLAM 算法室外动态场景中的表现，将其同先进的室外双目 SLAM 算法 DynaSLAM[7]、DOT-SLAM[21]、DynaSLAM Ⅱ[22] 进行对比，使用 KITTI 数据集[23] 中包含地面轨迹真值的双目图像序列 00-10 进行测试。

KITTI 数据集是一个广泛用于移动机器人和自动驾驶研究的基准测试集，由卡尔斯鲁厄理工学院和丰田汽车北美信息中心联合开发。其收录了多种真实世界的驾驶场景，包括城市、乡村和高速公路环境，提供了丰富的场景变化，可用于完成各种挑战性任务。

表 5 列出了各种算法 ATE 的 RMSE，本文算法在其中 7 个序列中表现为最优或次优，相比 ORB-SLAM3，本文算法在大多数序列表现更为出色，结果验证了本文算法能够有效提升室外动态场景下 SLAM 算法的跟踪精度，各序列的绝对轨迹误差减小 15%～25%。同时，在同 DynaSLAM、DOT-SLAM、DynaSLAM Ⅱ 的比较中发现，在图像序列 01、02 和 08 中，由于受到深度阴影和光照不均的影响，特征提取与匹配质量不佳，导致本文算法竞争力不足，除此之外，都能获得比较优秀的结果。

表 5 KITTI 数据集绝对轨迹误差的均方根误差结果

Table 5 Results of RMSE for ATE on KITTI dataset

m

序列	ORB-SLAM3[14]	DynaSLAM[7]	DOT-SLAM[21]	DynaSLAM Ⅱ[22]	本文算法
00	**0.96**	1.4	1.18	1.29	**0.96**
01	11.89	9.4	**1.33**	2.31	7.50
02	4.81	6.7	1.04	**0.91**	3.56
03	**0.29**	0.6	1.00	0.69	0.38
04	0.18	0.2	1.00	1.42	**0.15**
05	**0.39**	0.8	1.14	1.34	0.46
06	0.71	0.8	1.07	**0.19**	0.37
07	0.43	0.5	1.00	3.10	**0.41**
08	3.08	3.5	**1.24**	1.68	2.73
09	1.62	1.6	3.89	5.02	**1.06**
10	1.11	1.2	**1.02**	1.30	1.23

注：每个序列的最优结果用粗体突出显示。

3.3 消融实验

为系统评估各个模块对算法整体性能的影响，逐一去掉或修改 SLAM 算法中新增的模块，运行 TUM 数据集进行测试对比。其中，W/O-GMS 表示不区分场景的动态程度，全部视作高动态场景；W/O-MCC 表示室内高动态环境下，预先定义人类为唯一动态类，直接剔除动态框内的所有特征点；W/O-DAT 表示不进行目标检测和跟踪，在没有语

义先验的情况下, 只使用对极几何约束处理动态特征。

表 6 记录了上述 4 组不同配置下运行 TUM 数据集得到的 ATE 结果, 分析实验结果可知: W/O-GMS 对环境中物体的轻微运动容忍度不高, 将图像序列中所有帧视作高动态进行处理, 导致在低动态序列误差较大。W/O-MCC 对大部分的测试序列产生消极影响, 由于无法识别除人外的其他运动对象, 算法直接剔除人框内的全部特征点, 与人框重叠的其他框内点也被消除, 舍弃过多特征点使算法缺乏足够的特征点进行有效跟踪。W/O-DAT 在低动态场景表现优秀, 与之相反的是, 在高动态场景中, 由于对极几何约束通常基于场景中的大部分特征点为静态的假设, 缺少语义信息划分潜在动态区域, 无法有效区分所有动态特征点, 导致应对复杂的高动态场景的性能欠佳。

表 6　不同配置的绝对轨迹误差结果

Table 6　Results of ATE with different configurations　　m

序列	RMSE				S.D.			
	本文算法	W/O-GMS	W/O-MCC	W/O-DAT	本文算法	W/O-GMS	W/O-MCC	W/O-MCC
fr3/s/half	0.0196	0.056 5	0.031 6	**0.019 1**	0.0110	0.042 3	0.016 8	**0.008 7**
fr3/s/rpy	**0.020 6**	0.030 8	0.033 9	0.0212	**0.009 5**	0.020 9	0.025 0	0.0133
fr3/s/static	**0.006 3**	0.007 7	0.007 4	0.0068	**0.002 8**	0.004 0	0.003 4	0.0034
fr3/s/xyz	0.011 6	0.013 0	**0.009 3**	0.0104	0.007 2	0.006 7	**0.005 2**	0.0054
fr3/w/half	**0.018 9**	0.024 2	0.0234	0.027 9	**0.009 9**	0.014 0	0.0125	0.012 7
fr3/w/rpy	**0.034 2**	0.0415	0.044 9	0.054 6	**0.019 0**	0.0232	0.027 0	0.034 2
fr3/w/static	**0.006 3**	0.0082	0.008 3	0.014 7	**0.002 9**	0.0039	0.003 9	0.007 2
fr3/w/xyz	0.0135	**0.013 3**	0.014 2	0.019 2	0.0069	**0.006 8**	0.006 8	0.009 5

注: 每个序列的最优结果用粗体突出显示, 次优结果用下划线突出显示。

3.4　运行效率测试

除算法跟踪精度外, 算法的运行速度也至关重要, 为确保算法能即时响应场景变化, 从而满足实时运行的应用需求, 需要对算法运行效率进行测试。不同算法运行 TUM 数据集的每帧平均跟踪时间和所用硬件平台如表 7 所示, DynaSLAM 和 DS-SLAM 采用语义分割网络, 高计算成本使得跟踪时间较长。相比较而言, 本文算法基于 TensorRT 部署目标检测和跟踪算法, 保持模型精度的同时大幅提升了运行效率。与同样使用单阶段目标检测算法的 Crowd-SLAM、SG-SLAM 和 OVD-SLAM 相比, 本文算法提出了更为全面的动态特征抑制策略且进行目标检测缺失补偿, 由于算法复杂度有所增加, 时间成本略有增加。与 W/O-GMS 相比, 本文算法引入双类别动态场景处理策略, 快速过滤低动态帧中的误匹配, 有效提升低动态场景下算法整体运行效率。与原算法 ORB-SLAM3 相比, 本文算法增设了语义线程和稠密建图线程, 平均每帧要增加约 13 ms 的运行时间, 但仍然可以满足机器人的实时性需求。

表 7　不同算法的运行时间比较

Table 7　Runtime comparison of different systems

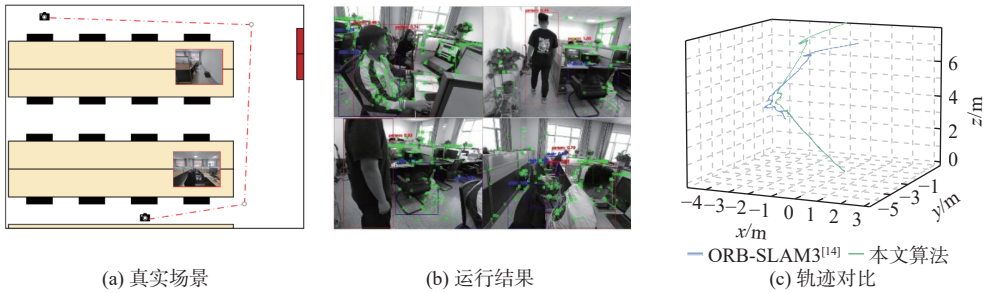
算法	每帧跟踪时间/ms	实验平台
本文算法	28.20～29.96	Intel i9, Nvidia RTX 4060 laptop
DynaSLAM	>350	CPU, Nvidia Tesla M40 GPU
DS-SLAM	59.4	Intel i7, P4000 GPU
ORB-SLAM3	20.24～22.79	Intel i9, Nvidia RTX 4060 laptop
ORB-SLAM3	27.61～29.77	Intel i9, Nvidia RTX 4060 laptop
ORB-SLAM3	20.68～23.35	Intel i9, Nvidia RTX 4060 laptop
W/O-GMS	30.22～32.67	Intel i9, Nvidia RTX 4060 laptop
ORB-SLAM3	14.32～17.58	Intel i9

3.5　真实场景测试

为进一步评估本文 SLAM 算法的实时性能, 在一个真实的实验室场景中设计一个包含不同动态程度的实时测试。实验室面积约为 8 m×6 m。手持 Intel RealSense D455 相机以分辨率为 640×480 和帧率为 30 帧/s 获取输入数据。

在实验过程中, 手动旋转并移动相机, 在房间里手持相机行走。图 10(a) 为步行路线俯视图, 红色虚线表示相机的运动轨迹。图 10(b) 可以观察到有人在镜头前行走, 其身上的动态特征点被精确识别并剔除。还有人坐在椅子上交谈, 做着随机的动作, 其中处于静止状态的人身上的特征点得以有效保留。运行结果表明, 本文 SLAM 算法根据实时输入快速响应, 准确判定物体的运动状态, 并对存在的动态特征进行针对性处理, 相机轨迹跟踪稳定, 在真实场景下具有良好的实时性。

图 10(c) 中 ORB-SLAM3 的估计轨迹在高动态



(a) 真实场景 (b) 运行结果 (c) 轨迹对比

图 10 真实场景实验

Fig. 10 Real scene experiment

场景中出现了明显的漂移，而本文算法的估计轨迹与真实轨迹更为相似，这一结果证明本文算法能够提升相机位姿估计精度，有效应对动态环境带来的挑战。

4 结 论

1）本文提出一种面向动态场景实时视觉SLAM算法，在室内和室外环境均可有效运行，适配单目、双目、RGB-D多种相机，能够有效处理一般性场景中有限的动态物体。

2）本文算法利用多目标跟踪算法对YOLOX-s获取的语义信息进行缺漏检补偿，实现目标检测算法效率与精度的平衡。

3）本文算法基于图像帧间特征匹配精度划分场景的动态程度，增强其在复杂动态场景下的适应能力。在低动态场景中，利用GMS算法快速筛选并剔除误匹配，节约大量计算成本；在高动态场景中，引入卡方检验进行运动一致性检测，不依赖语义先验识别动态异常点，设置自适应阈值判定检测框运动属性，有效剔除动态特征点，同时最大限度地保留静态特征点，以增加稳定特征点的数量，实现鲁棒的高精度跟踪。

4）本文算法使用公开数据集TUM、BONN及KITTI进行性能评估，实验结果表明，本文算法在动态环境中的综合性能优于大多数现有其他算法。此外，真实场景测试进一步证明本文算法在现实环境中的有效性和实时性。

参考文献（References）

[1] 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.

[2] 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.

[3] Klein G, Murray D. Parallel tracking and mapping for small AR workspaces[C]//Proceedings of the 2007 6th IEEE and ACM International Symposium on Mixed and Augmented Reality. Piscataway: IEEE Press, 2007: 225-234.

[4] Cheng J Y, Wang C Q, Meng M Q H. Robust visual localization in dynamic environments based on sparse motion removal[J]. IEEE Transactions on Automation Science and Engineering, 2020, 17(2): 658-669.

[5] Dai W C, Zhang Y, Li P, et al. RGB-D SLAM in dynamic environments using point correlations[J]. IEEE Transactions on Pattern Analysis and Machine Intelligence, 2022, 44(1): 373-389.

[6] Du Z J, Huang S S, Mu T J, et al. Accurate dynamic SLAM using CRF-based long-term consistency[J]. IEEE Transactions on Visualization and Computer Graphics, 2022, 28(4): 1745-1757.

[7] 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.

[8] Yu C, Liu Z X , Liu X J, et al. DS-SLAM: a semantic visual SLAM towards dynamic environments[C]//Proceedings of the 2018 IEEE/RSJ International Conference on Intelligent Robots and Systems. Piscataway: IEEE Press, 2018: 1168-1174.

[9] 王浩，卢德玖，方宝富. 动态环境下基于增强分割的RGB-D SLAM方法[J]. 机器人, 2022, 44(4): 418-430.
Wang H, Lu D J, Fang B F. RGB-D SLAM method based on enhanced segmentation in dynamic environment[J]. Robot, 2022, 44(4): 418-430 (in Chinese).

[10] 齐浩，付悦欣，胡祝华，等. 基于自适应阈值和速度优化的轻量化语义VSLAM方法[J]. 北京航空航天大学学报, 2025, 51(7): 2562-2572.
Qi H, Fu Y X, Hu Z H, et al. A lightweight semantic VSLAM approach based on adaptive thresholding and speed optimization[J]. Journal of Beijing University of Aeronautics and Astronautics, 2025, 51(7): 2562-2572(in Chinese).

[11] Soares J C V, Gattass M, Meggiolaro M A. Crowd-SLAM: visual SLAM towards crowded environments using object detection[J]. Journal of Intelligent & Robotic Systems, 2021, 102(2): 50.

[12] Cheng S H, Sun C H, Zhang S J, et al. SG-SLAM: a real-time RGB-D visual SLAM toward dynamic scenes with semantic and geometric information[J]. IEEE Transactions on Instrumentation and Measurement, 2023, 72: 7501012.

[13] He J M, Li M R, Wang Y Y, et al. OVD-SLAM: an online visual SLAM for dynamic environments[J]. IEEE Sensors Journal, 2023, 23(12): 13210-13219.

[14] Campos C, Elvira R, Rodriguez 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.

[15] Ge Z, Liu S T, Wang F, et al. YOLOX: exceeding YOLO series in 2021[EB/OL]. V2. arXiv (2021-08-06)[2021-08-06]. https://doi.org/10.48550/arXiv.2107.08430.

[16] Aharon N, Orfaig R, Bobrovsky B Z. BoT-SORT: robust associations multi-pedestrian tracking[EB/OL]. (2022-07-07)[2022-07-26]. https://doi.org/10.48550/arXiv.2206.14651.

[17] Bian J W, Lin W Y, Liu Y, et al. GMS: grid-based motion statistics for fast, ultra-robust feature correspondence[J]. International Journal of Computer Vision, 2020, 128(6): 1580-1593.

[18] Liu G H, Zeng W L, Feng B, et al. DMS-SLAM: a general visual SLAM system for dynamic scenes with multiple sensors[J]. Sensors, 2019, 19(17): 3714.

[19] Sturm J, Engelhard N, Endres F, et al. A benchmark for the evaluation of RGB-D SLAM systems[C]//Proceedings of the 2012 IEEE/RSJ International Conference on Intelligent Robots and Systems. Piscataway: IEEE Press, 2012: 573-580.

[20] Palazzolo E, Behley J, Lottes P, et al. ReFusion: 3D reconstruction in dynamic environments for RGB-D cameras exploiting residuals [C]//Proceedings of the 2019 IEEE/RSJ International Conference on Intelligent Robots and Systems. Piscataway: IEEE Press, 2019: 7855-7862.

[21] Ballester I, Fontán A, Civera J, et al. DOT: dynamic object tracking for visual SLAM[C]//Proceedings of the 2021 IEEE International Conference on Robotics and Automation. Piscataway: IEEE Press, 2021: 11705-11711.

[22] 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.

[23] Geiger A, Lenz P, Urtasun R. Are we ready for autonomous driving? The KITTI vision benchmark suite[C]//Proceedings of the 2012 IEEE Conference on Computer Vision and Pattern Recognition. Piscataway: IEEE Press, 2012: 3354-3361.

A real-time visual SLAM algorithm for complex dynamic scenarios

ZHAO Boting[1]，LIU Jun[1,*]，CHAI Hongxu[2]，ZHANG Jianye[1]，WU Ruibin[1]

(1. School of Mechanical and Electrical Engineering，Lanzhou University of Technology，Lanzhou 730050，China；
2. Gansu Province Special Equipment Inspection and Testing Institute，Lanzhou 730050，China)

Abstract：Simultaneous localization and mapping (SLAM) systems frequently have a number of issues in general or sophisticated dynamic scenarios with dynamic objects, including strong assumptions, inadequate positioning precision, or even failure. To solve these problems, a real-time visual SLAM algorithm for complex dynamic scenarios is proposed, which is built on ORB-SLAM3. The algorithm deploys YOLOX-s based on accelerated model inference to obtain semantic prior information to process dynamic objects, and the Kalman filter and Hungarian algorithm are combined to compensate for missing object detection. A two-category dynamic scenario processing strategy is proposed based on the grid-based motion statistics (GMS) feature matching algorithm to rapidly screen reliable matching points, which separates the scenarios into two categories based on the image matching accuracy, in order to enhance the positioning accuracy and generalization ability of the algorithm. For low dynamic frames, the GMS algorithm is used to quickly filter out dynamic features, so as to avoid a large number of matrix operations and improve operational efficiency. For high dynamic frames, the Chi-square test is utilized to identify motion outliers, and a dynamic feature culling strategy is proposed to cull dynamic points according to the adaptive threshold and keep the tracking stability. The experimental results on standard datasets and real scenarios show that the algorithm effectively addresses the real-time localization challenges, demonstrating its superior accuracy, robustness and real-time performance in complex indoor and outdoor dynamic scenarios.

Keywords：visual SLAM; dynamic object; deep learning; object detection; scenario classification

Received：2024-09-05；**Accepted**：2024-11-07；**Published Online**：2024-12-03 12：14

URL：link.cnki.net/urlid/11.2625.V.20241203.0904.002

Foundation items：National Natural Science Foundation of China (71861025); Independent Research and Development Project of Gansu Provincial Department of Science and Technology (22YF11GA315)；Science and Technology Support Project of Lanzhou (2024-8-46)

* **Corresponding author**. E-mail：lzhjliu@126.com