Software Engineering and Applications 软件工程与应用, 2023, 12(2), 366-380
Published Online April 2023 in Hans. https://www.hanspub.org/journal/sea
https://doi.org/10.12677/sea.2023.122037

Hans汉斯

基于相机与激光雷达传感器融合的SLAM方法

王睿忠，汤玉春

上海理工大学光电信息与计算机学院，上海

收稿日期：2023年2月13日；录用日期：2023年4月20日；发布日期：2023年4月29日

摘　要

针对SLAM系统中动态物体对机器人定位和建图造成的误差，提出将相机和激光雷达传感器数据进行融合的方法。采用DBSCAN非监督聚类方法对激光雷达点云进行聚类并分割，对分割后的物体采用改进的3D点云簇评分和匈牙利算法进行匹配，实现相邻帧之间的同一物体判别和物体运动状态的判断；通过统一相机和激光雷达坐标，将属于动态物体范围内的特征点滤除，达到提高后端位姿估计精度的效果。实验结果表明：在经过该融合后的SLAM系统，机器人的旋转误差和平移误差有了明显的下降，定位精度和建图效果得到较好提升。

关键词

点云聚类，SLAM，机器人，动态物体，相机，激光雷达

Fusion Method Based on Camera and Lidar for SLAM

Ruizhong Wang, Yuchun Tang

School of Optical-Electrical and Computer Engineering, University of Shanghai for Science and Technology, Shanghai

Received: Feb. 13th, 2023; accepted: Apr. 20th, 2023; published: Apr. 29th, 2023

Abstract

Aiming at the errors caused by dynamic objects in the robot positioning and mapping in the SLAM system, a method of fusing camera and lidar sensor data is proposed. The DBSCAN unsupervised clustering method is used to cluster and segment the lidar point cloud, and the segmented object is matched with the improved 3D point cloud cluster score and the Hungarian algorithm to achieve similar Discriminate the same object between adjacent frames and judge the motion state of the object; by unifying the coordinates of the camera and lidar, the feature points belonging to

文章引用: 王睿忠, 汤玉春. 基于相机与激光雷达传感器融合的 SLAM 方法[J]. 软件工程与应用, 2023, 12(2): 366-380.
DOI: 10.12677/sea.2023.122037

the dynamic object range are filtered out to improve the accuracy of the back-end pose estimation. The experimental results show that after the fused SLAM system, the rotation error and translation error of the robot have been significantly reduced, and the positioning accuracy and mapping effect have been better improved.

Keywords

Point Cloud, SLAM, Robot, Dynamic Object, Camera, Lidar

Copyright © 2023 by author(s) and Hans Publishers Inc.
This work is licensed under the Creative Commons Attribution International License (CC BY 4.0).
http://creativecommons.org/licenses/by/4.0/
Open Access

1. 引言

随着近年来各类应用机器人、无人飞行器和自动驾驶汽车的发展，越来越智能化的机器人已经成为现实生活中的一部分。同时定位与地图构建(Simultaneous Localization and Mapping, SLAM)技术早期起源于机器人的发展，为其自动化的工作状态提供稳定的位姿信息。SLAM 在实现移动机器人的自我估计位姿和建立环境关系地图有着强大能力，定位技术与建图技术相辅相成。基于相机传感器的视觉 SLAM，通过连续的相机成像，跟踪设置关键点，并采用三角化、PnP 或者 ICP 方法完成周围地图点的坐标计算，并通过地图点的坐标变化推测相机的姿态变化。基于激光雷达的激光 SLAM 使用点云匹配方法或改进的匹配方法，计算出两帧点云之间的旋转和平移信息，其精度可达厘米级别。但是，基于相机的视觉 SLAM 能够提供丰富的图像信息，但其往往也对光照的变化和物体颜色变化非常敏感；基于激光雷达的激光 SLAM 系统可以在依赖周围环境光照的情况下工作，但其所能提供的信息相对较为单调，无法满足用户对于颜色或类别上的判断。

在过去的 SLAM 发展中，无论视觉 SLAM 还是激光 SLAM 算法都是在假设周围物体为静止状态下，通过计算周围静止路标点在两帧图像或者点云中的变化来推算出机器人自身的位姿变化。如何剔除周围环境中动态路标点的影响是提升 SLAM 算法精确性的关键。在单一传感器的 SLAM 系统中，不少学者提出了解决动态物体影响的方法，例如，Yu C, Liu Z, Liu X [1]等人提出通过使用语义分割方法对语义对象进行一致性检测，去除运动语义对象中的关键点，剔除运动关键点对 SLAM 系统的影响；Brsan I A, Liu P, Pollefeys M [2]等提出使用双目相机实现机器人位姿计算和构建地图，并对左目相机的图像进行语义分割后根据语义提示和稀疏场景流将周围环境区分为动静态物体，在不丢弃动态物体的情况下实现对机器人定位与建图；Chen X, Li S, Mersch B [3]等人通过将激光雷达点云转换为残差图像，依靠卷积神经网络实现动静态物体的分割。

本文采用将对激光雷达点云进行 DBSCAN [4]聚类，判断出聚类物体中的动态物体，对处于动态物体范围内的相机特征点进行剔除，实现相机与激光雷达信息的有效融合，达到提高机器人定位与建图精度。

2. 基于 DBSCAN 的激光点云聚类和目标追踪

2.1. DBSCAN 聚类算法

基于密度的空间聚类(DBSCAN)算法是基于密度连通区域的非监督聚类方法[5]，算法使用两个参数

Eps 和 MinPts 用来描述邻域的样本分布紧密程度。其中 Eps 是某一样本的邻域距离阈值，MinPts 是某一样本的距离为 Eps 的邻域中样本个数阈值。由密度可达关系导出的最大密度相连的样本集合，最终聚类为一个簇。算法相关定义如下：

Eps(ϵ)-邻域：对于 $x_j \in D$，其 ϵ -邻域包含样本集 D 中与 x_j 的距离不大于 ϵ 的子样本集，即 $N \in (x_j) = \{x_i \in D \mid \mathrm{dist}(x_i, x_j) \le \epsilon\}$，该子样本集的个数记为 $|N \in (x_j)|$；

核心对象：对于任一样本 $x_j \in D$，如果其 ϵ -邻域对应的 $N \in (x_j)$ 至少包含 MinPts 个样本，即如果 $|N \in (x_j)| \ge \mathrm{MinPts}$，则 x_j 是核心对象。

密度直达：如果 x_i 位于 x_j 的 ϵ -邻域中，且 x_j 是核心对象，则称 x_i 由 x_j 密度直达。

密度可达：对于 x_i 和 x_j，如果存在样本序列 $p_1, p_2, \cdots, p_T$，满足 $p_1 = x_i$，$p_T = x_j$，且 p_{T+1} 由 p_T 密度直达，则称 x_j 由 x_i 密度可达。

密度相连：对于 x_i 和 x_j，如果存在核心对象样本 x_k，使 x_i 和 x_j 均由 x_k 密度可达，则称 x_i 和 x_j 密度相连。

DBSCAN 从任意点 p 开始，从数据集中检索所有密度可达的点，Eps 和 MinPts。如果 p 是一个核心点，该过程即可生成一个新的点云簇，Eps 和 MinPts [6]；如果 p 是边界点，那么 p 没有密度可达的点，DBSCAN 访问数据集中的下一个点。当两个不同密度点云簇 S_1, S_2 的距离 $\mathrm{dist}(S_1, S_2) = \min\{\mathrm{dist}(p,q) \mid p \in S_1, q \in S_2\}$ 小于全局 Eps 时，可将两个点云簇合并为同一个点云簇。递归调用 DBSCAN，直到所有的数据点都被访问，不再有新的点加入到任何簇中，则算法结束。算法过程见表 1：

Table 1. Process of DBSCAN algorithm
表 1. DBSCAN 算法过程

算法 1.DBSCAN 算法
输入：包含 n 个样本点的数据集 $D = (x_1, x_2, \cdots, x_n)$；邻域参数(Eps, MinPts)
输出：点云簇划分 C

1) 初始化核心对象集合 $\Omega = \varnothing$，初始化聚类簇数 $k = 0$，初始化未访问样本集合 $\Gamma = D$，簇划分 $C = \varnothing$
2) 对于 $j = 1, 2, \cdots, n$，按以下步骤寻找出其核心对象：
a) 通过距离度量方式找到样本 x_j 的 ϵ 邻域子样本集 $N_\epsilon(x_j)$
b) 如果子样本集个数满足 $|N_\epsilon(x_j)| \ge \mathrm{MinPts}$，将样本 x_j 加入核心对象样本集合：$\Omega = \Omega \cup \{x_j\}$
3) 如果核心对象集合 $\Omega = \varnothing$，则算法结束，否则转入步骤 4。
4) 在核心对象 Ω 中，随机选择一个核心对象 o，初始化当前簇核心对象队列 $\Omega_{cur} = \{o\}$，初始化类别序号 $k = k + 1$，初始化当前簇样本集合 $C_k = \{o\}$，更新未访问样本集 $\Gamma = \Gamma - \{o\}$
5) 如果当前簇核心对象队列 $\Omega_{cur} = \varnothing$，则当前聚类簇 C_k 生成完毕，更新簇划分 $C = (C_1, C_2, \cdots, C_k)$，更新核心对象 $\Omega = \Omega - C_k$，转入步骤 3。
6) 在当前簇核心对象队列 Ω_{cur} 中取出一个核心对象 o'，通过邻域距离阈值 Eps 找出所有 Eps-邻域子样本集 $N_\epsilon(o')$，令 $\Delta = N_\epsilon(o') \cap \Gamma$，更新当前簇类样本集合 $C_k = C_k \cup \Delta$，更新未访问样本集合 $\Gamma = \Gamma - \Delta$，更新 $\Omega_{cur} = \Omega_{cur} \cup (\Delta \cap \Omega) - o'$，转入步骤 5。
7) 输出簇划分：$C = (C_1, C_2, \cdots, C_k)$

2.2. 基于距离、方向和激光点数目的目标评分计算

1) 基于距离比的目标点云簇评分。经过 2.1 中的 DBSCAN 聚类，得到前后帧的目标点云簇

$C_{\text{pre_frame}}=(C_1,C_2,\cdots,C_m)$，$C_{\text{cur_frame}}=(C_1',C_2',\cdots,C_n')$，采用均值计算方法求出对应点云簇的中心点点云坐标$(o_1,o_2,\cdots,o_m)$和$(o_1',o_2',\cdots,o_n')$，则前后帧点云簇的距离可表示为：

$$\text{dist}(C_i,C_j')=\left\{\text{dist}(o_i,o_j')\mid o_i\in C_i,o_j'\in C_j',i=1,2,\cdots,m;j=1,2,\cdots,n\right\} \tag{1}$$

定义$\text{dist}_{\text{threshold}}$，当$\text{dist}(C_i,C_j')>\text{dist}_{\text{threshold}}$时，$\text{score}_{\text{dist}}(C_i,C_j')=0$，否则，$\text{score}_{\text{dist}}(C_i,C_j')=\dfrac{\text{dist}(C_i,C_j')}{\text{dist}_{\text{threshold}}}$。

2) 基于方向的目标点云簇评分。机器人在运动过程中，周围物体点云簇中心与机器人自身中心点所形成的角度具有很强的辨识性，通过增加中心点的方向性评分可以有效弥补(1)中基于距离评分所带来的无向性。即，定义o为机器人自身中心，$\text{angle}(o,o_i)$表示o_i到o点的夹角，则：

$$\text{direction}(O,C_i)=\left\{\text{angle}(o,o_i)\mid o_i\in C_i,i=1,2,\cdots,m\right\} \tag{2}$$

$$\text{direction}(O,C_j')=\left\{\text{angle}(o,o_j')\mid o_j'\in C_j',j=1,2,\cdots,n\right\} \tag{3}$$

$$\begin{cases}\text{score}_{\text{direction}}(C_i,C_j')=\dfrac{\frac{\pi}{4}-\text{direction}(C_i,C_j')}{\frac{\pi}{4}}=\dfrac{\frac{\pi}{4}-\left[\text{angle}(o,o_i)-\text{angle}(o,o_j')\right]}{\frac{\pi}{4}}\\ \text{score}_{\text{direction}}(C_i,C_j')=0,\ \text{if}\left|\text{angle}(o,o_i)-\text{angle}(o,o_j')\right|>\dfrac{\pi}{4}\end{cases} \tag{4}$$

3) 基于激光点数量对比的点云簇评分。在正常的运动速度下，激光雷达对同一个物体的探测面积在较短时间内不会发生较大变化，即同一个物体在相邻两帧的激光点数量上不会出现较大的差异。以激光点数量的相似性可以判较小的时间间隔下两个点云簇的相似程度，因此，以用点云数目的比例作为两个物体相似性的评分可计算为：

$$\text{score}_{\text{num}}(C_i,C_j')=\frac{\min\left(\text{num}(C_i),\text{num}(C_j')\right)}{\max\left(\text{num}(C_i),\text{num}(C_j')\right)} \tag{5}$$

综合(1)，(2)，(3)点云簇的相似性评分可表示为距离评分与数目评分的均值并与方向评分的乘积，即，

$$\text{score}(C_i,C_j')=\left\{\left[1-\text{score}_{\text{dist}}(C_i,C_j')\right]*\frac{1}{2}+\text{score}_{\text{num}}(C_i,C_j')*\frac{1}{2}\right\}*\text{score}_{\text{direction}}(C_i,C_j') \tag{6}$$

$$\text{s.t.}\begin{cases}\text{score}_{\text{dist}}(C_i,C_j')\le 1\\ \text{score}_{\text{num}}(C_i,C_j')\le 1\\ \text{score}_{\text{direction}}(C_i,C_j')\le 1\\ i=1,2,\cdots,m;j=1,2,\cdots,n\end{cases} \tag{7}$$

算法流程见表 2：

Table 2. Points cluster similarity scoring algorithm
表 2. 点云簇相似性评分算法

算法 2.点云簇相似性评分算法
输入：前后帧点云簇 $C_{\text{pre_frame}}=\left(C_1,C_2,\cdots,C_m\right)$， $C_{\text{cur_frame}}=\left(C_1',C_2',\cdots,C_n'\right)$；点云簇中心距离值 $\text{dist}_{\text{threshold}}$
输出：点云簇相似性评分 $\text{score}\left(C_i,C_j'\right)$

1)for $o_i\in C_i$ in $C_{\text{pre_frame}}$
2) for $o_j'\in C_j'$ in $C_{\text{cur_frame}}$
3) if $\text{dist}\left(o_i,o_j'\right)>\text{dist}_{\text{threshold}}$
4) $\text{score}_{\text{num}}\left(C_i,C_j'\right)=0$
5) endif
6) $\text{score}_{\text{num}}\left(C_i,C_j'\right)=\dfrac{\text{dist}\left(C_i,C_j'\right)}{\text{dist}_{\text{threshold}}}$
7) $\text{score}_{\text{dist}}\left(C_i,C_j'\right)=\dfrac{\min\left(\text{num}\left(C_i\right),\text{num}\left(C_j'\right)\right)}{\max\left(\text{num}\left(C_i\right),\text{num}\left(C_j'\right)\right)}$
8) $\text{score}\left(C_i,C_j'\right)=\left(1-\text{score}_{\text{dist}}\left(C_i,C_j'\right)\right)*\dfrac{1}{2}+\text{score}_{\text{num}}\left(C_i,C_j'\right)*\dfrac{1}{2}$
9) endfor
10) endfor

2.3. 基于目标评分的匈牙利匹配和目标追踪

匈牙利算法是由匈牙利数学家 Edmons 于 1965 年提出，算法的核心在于寻找增广路径，用增广路径求二分图的最大匹配。当前在机器视觉和图像处理研究中，匈牙利算法常用于解决帧与帧之间多个目标的匹配问题。本文通过使用 DBSCAN 聚类后的点云簇和对点云簇的相似性评分可转换为有权二部图的顶点权重，即两帧之间的点云簇匹配问题可变成采用匈牙利匹配的同时考虑总权重最大的问题或者总损失最小的问题，也就是建立前后两帧点云簇之间的最优匹配。建立数学模型如下：设 $S_{ij}>0\left(i=1,2,\cdots,m;j=1,2,\cdots,n\right)$ 表示点云簇集合 $\left(C_1,C_2,\cdots,C_m\right)$ 中第 i 簇和集合 $\left(C_1',C_2',\cdots,C_n'\right)$ 中第 j 簇匹配时的权重，决策变量 Z_{ij} 为：

$$Z_{ij}=\begin{cases}1, & \text{第}\,i\,\text{簇与第}\,j\,\text{簇匹配}\\ 0, & \text{第}\,i\,\text{簇与第}\,j\,\text{簇不匹配}\end{cases}\tag{8}$$

则匹配问题即可变为：

$$\max Z=\sum_{i=1}^{m}\sum_{j=1}^{n}S_{ij}Z_{ij}\tag{9}$$

$$\text{s.t.}\begin{cases}\sum_{i=1}^{m}Z_{ij}=1, & i=1,2,\cdots,m\\ \sum_{j=1}^{n}Z_{ij}=1, & j=1,2,\cdots,n\\ Z_{ij}=1\text{或}0, & i=1,2,\cdots,m;j=1,2,\cdots,n\end{cases}\tag{10}$$

$$\text{Matched}\left(C_i,C_j'\right)=\min\left(\text{Cost}\left(C_i,C_j'\right)\right)=1-\max Z=1-\sum_{i=1}^{m}\sum_{j=1}^{n}S_{ij}Z_{ij}\tag{11}$$

算法流程见表 3：

Table 3. Matching method using Hungarian algorithm
表 3. 匈牙利匹配算法

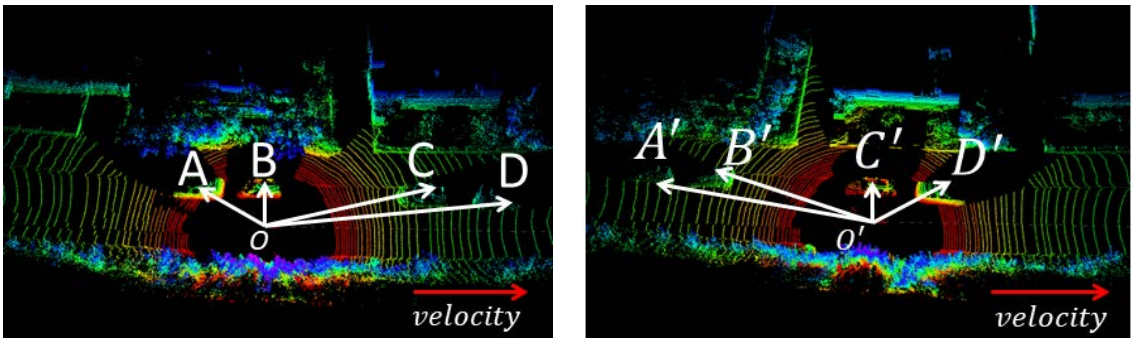
算法 3. 匈牙利匹配算法

输入：损失矩阵 $\mathrm{Cost}\left(C_i, C_j'\right) = 1 - \sum_{i=1}^{m} \sum_{j=1}^{n} S_{ij} Z_{ij}$

输出：匹配矩阵 $\mathrm{Matched}\left(C_i, C_j'\right)$

1) 在给定的损失矩阵中，遍历该行的所有列元素并减去该行成本最低的元素。确保每一行至少有一个零
2) 从第 1 步生成的结果成本矩阵中，遍历各列并减去每一列中成本最低的元素，确保每一列至少包含一个零
3) 分配零
a) 检查每一行的各个元素，找出各行中并未标记为零的元素。标记该元素并划掉与该元素同一列的其他元素。对其他行执行相同的操作
b) 检查每一列的各个元素，找出各列中并未标记为零的元素。标记该元素并划掉与该元素同一行的其他元素。对其他列执行相同的操作
4) 执行最佳测试
a) 如果每一行和每一列都恰好有一个标记元素，则当前分配是最优的
b) 如果至少一行或一列缺少分配(即，如果至少一行或一列缺少一个带标记的零元素)，则当前分配不是最佳的。继续执行步骤 5。从步骤 1 中创建的成本矩阵的每一列中的所有条目中减去成本最低的元素，并确保每一列至少有一个零
5) 绘制最少数量的直线以覆盖所有零点
a) 突出显示未分配的行；
b) 在已标记的行中对未标记的列进行标记
c) 在已标记的列中对未标记的行进行标记
d) 继续(b)和(c)，直到找不到新的未标记的行和列为止
e) 在所有未被标记的行和列上画线，得到覆盖所有零元素的最少直线数
6) 找出未被直线覆盖的最低成本元素。在所有未覆盖的元素中减去这个成本最低的元素，并将该最小元素添加到这些直线相交处的所有元素，得到新的损失矩阵
7) 继续第 1~6 步，直到找到最合适的分配
8) 输出匹配矩阵

图 1 为采用算法匹配后的前后帧点云簇匹配示例，在自车从 o 点到 o' 运动时，A、B、C、D 物体通过距离、方向和激光点数量的评分后，最终实现 AA′，BB′，CC′，DD′之间的一一匹配。



(a) 前一帧　　(b) 当前帧

Figure 1. Diagram of object point cloud matching in front and rear frames
图 1. 前后相邻帧物体点云匹配示意图

3. 激光雷达点云处理

激光雷达根据光的传播时间和光束发生器的角度可以计算出目标点云的距离从而得到点云的极坐

标，自然地，目标点云的空间欧式坐标即可由其极坐标计算获得。因为，目标点云在物理空间中的位置是固定的，那么该点云位于相机坐标系下的坐标和位于雷达坐标系下的坐标可以通过摄像头和激光雷达的旋转和平移将两种传感器坐标系统一到同一个坐标系下。整个过程可以分为旋转(Rotation)、平移(Translation)和缩放(Scaling) [7] [8]：

1) 将雷达坐标系旋转到相机坐标系下。

将雷达坐标绕 Z 轴旋转后的坐标为：

$$\begin{bmatrix} X_L' \\ Y_L' \\ Z_L' \end{bmatrix} = \begin{bmatrix} 1 & 0 & 0 \\ 0 & \cos\alpha & -\sin\alpha \\ 0 & \sin\alpha & \cos\alpha \end{bmatrix} \begin{bmatrix} X_L \\ Y_L \\ Z_L \end{bmatrix} \tag{12}$$

同理，将雷达坐标绕 X, Y 轴旋转之后的坐标为：

$$\begin{bmatrix} X_L' \\ Y_L' \\ Z_L' \end{bmatrix} = \begin{bmatrix} \cos\beta & 0 & \sin\beta \\ 0 & 1 & 0 \\ -\sin\beta & 0 & \cos\beta \end{bmatrix} \begin{bmatrix} X_L \\ Y_L \\ Z_L \end{bmatrix} \tag{13}$$

$$\begin{bmatrix} X_L' \\ Y_L' \\ Z_L' \end{bmatrix} = \begin{bmatrix} \cos\gamma & -\sin\gamma & 0 \\ \sin\gamma & \cos\gamma & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} X_L \\ Y_L \\ Z_L \end{bmatrix} \tag{14}$$

合并以上三次操作得到坐标系的旋转矩阵 R，

$$R = \begin{bmatrix} \cos\alpha\cos\beta & -\sin\alpha\cos\beta & \sin\beta \\ \sin\alpha\sin\beta\cos\gamma + \cos\alpha\sin\gamma & \cos\alpha\cos\gamma - \sin\alpha\sin\beta\sin\gamma & -\sin\alpha\cos\beta \\ \sin\alpha\sin\gamma + \cos\alpha\cos\beta\cos\gamma & \cos\alpha\sin\beta\sin\gamma + \sin\alpha\cos\gamma & \cos\alpha\cos\beta \end{bmatrix} \tag{15}$$

2) 通过平移使雷达坐标原点与相机坐标系原点重合。

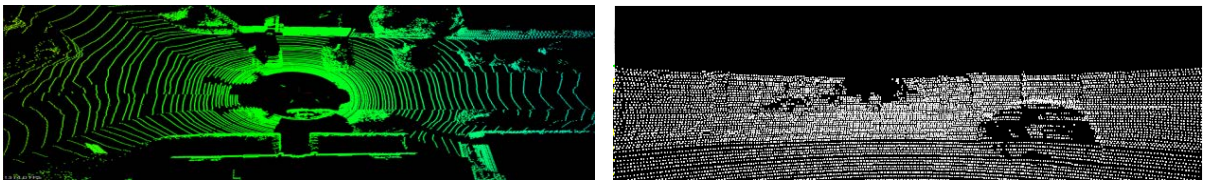
假设两个坐标系原点在坐标轴上的偏差用向量表示为 $t = [x, y, z]^{\mathrm{T}}$，雷达观测到目标点坐标转换到相机坐标系下的坐标就可以表示为[9]，

$$\begin{bmatrix} X_c \\ Y_c \\ Z_c \\ 1 \end{bmatrix} = \begin{bmatrix} R & t \\ 0 & 1 \end{bmatrix} \begin{bmatrix} X_l \\ Y_l \\ Z_l \\ 1 \end{bmatrix} \tag{16}$$

在已知相机的内参矩阵 $K = \begin{bmatrix} f_x & 0 & c_x \\ 0 & f_y & c_y \\ 0 & 0 & 1 \end{bmatrix}$ 的情况下，目标点点云坐标从雷达坐标到图像坐标的转换可表示为：

$$Z \begin{bmatrix} u \\ v \\ 1 \end{bmatrix} = \begin{bmatrix} f_x & 0 & c_x \\ 0 & f_y & c_y \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} X_c \\ Y_c \\ Z_c \end{bmatrix} \tag{17}$$

综上，激光雷达点云的坐标变换可以通过旋转和平移变换得到相机坐标系坐标[10]，并经过缩放后显示到图像平面上。如图 2，图 2(a)的原始点云通过坐标系变换得到图 2(b)像相机坐标系下的点云视图。



(a) 原始点云视图　　(b) 转换到相机坐标系下的点云视图

Figure 2. Rendering of lidar point cloud conversion
图 2. 激光雷达点云转换效果图

4. 融合算法

4.1. 运动物体判断

运动中的机器人第 k 时刻的雷达观测到目标物体 P 的中心点坐标 $P_k^i=\left(x_k^i,y_k^i,z_k^i\right)$，第 $k+1$ 时刻观测的该物体坐标为 $P_{k+1}^i=\left(x_{k+1}^i,y_{k+1}^i,z_{k+1}^i\right)$，本文已在第一节中实现了 P_k^i 和 P_{k+1}^i 的匹配，那么判断 P 是否为运动目标的问题将转换为与整体运动状态是否一致的问题。因此，可将雷达帧中各物体的坐标都统一缩放到半径为 1 的坐标平面上，并假设 k 时刻雷达的位姿为 T_k，那么雷达在 $k+1$ 时刻的位姿 T_{k+1} 就可以通过物体 P 在第 k 和第 $k+1$ 帧点云的位姿相对变化 $\Delta T_{k,k+1}$ 求得，即：

$$T_{k+1}=T_k\Delta T_{k,k+1} \tag{18}$$

在 P 点未发生运动的情况下，P_k^i 和 P_{k+1}^i 通过各自的变换矩阵变换转得到相同的世界坐标系，即：

$$P_w^i=\begin{bmatrix}x_w^i\\y_w^i\\z_w^i\end{bmatrix}=T_k\begin{bmatrix}x_k^i\\y_k^i\\z_k^i\end{bmatrix}=T_{k+1}\begin{bmatrix}x_{k+1}^i\\y_{k+1}^i\\z_{k+1}^i\end{bmatrix} \tag{19}$$

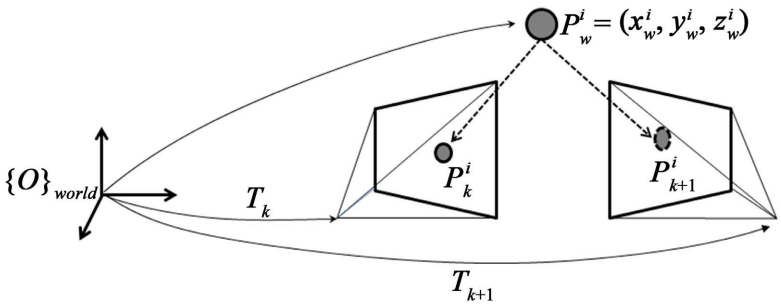


Figure 3. Position relationship of the same point in adjacent frames
图 3. 相邻帧中同一点的位置关系

如图 3，当物体处于 P_w^i 静止状态下时，其相邻帧所计算该点的世界坐标必然是相同的。取 $\delta_{k,k+1}^i=T_kP_k^i-T_{k+1}P_{k+1}^i, i\in(1,N)$，则：

$$\delta_{k,k+1}^i=T_k\left(P_k^i-\Delta T_{k,k+1}P_{k+1}^i\right) \tag{20}$$

式中 T_k 和 $\Delta T_{k,k+1}$ 对于同一帧内的各个物体都具有相同的效果，其在影响数值的大小外并不影响具体值是否偏离于均值的判断，因而可用单位矩阵来予以代替。

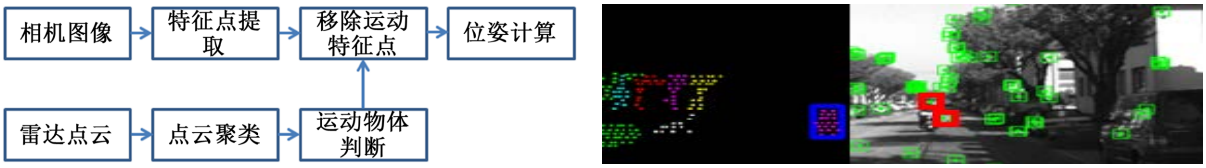
$$\bar{\delta}=\frac{1}{N}\sum_{j=1}^{N}\delta_{k,k+1}^j=\frac{1}{N}\sum_{j=1}^{N}\left(P_k^i-P_{k+1}^i\right) \tag{21}$$

当 $\delta_{k,k+1}^i\neq\bar{\delta}$，则 P_w^i 可认定为运动点，即对应的 P_k^i 和 P_{k+1}^i 为运动点在 K 时刻和 K+1 时刻的投影。

4.2. 数据融合

在第一节中，采用算法 1、2、3 实现了相邻帧物体点云簇的追踪，通过 4.1 的方法实现对运动物体的判断，第二节中的方法实现对相机传感器和雷达传感器感知物体坐标统一到统一坐标系下，为视觉 SLAM 和激光 SLAM 的融合提供了实现途径。

基于特征点法的 SLAM 系统通过前端对相机拍摄的图片提取特征点，并通过后端将提取的特征点进行计算获得相机的位姿，由于特征点法在特征提取的过程中对动态物体的不敏感性导致会将属于运动的物体的特征点纳入位姿计算中，从而引起位姿估算上的偏差。本文使用对激光雷达点云判断出的运动物体坐标进行标记，将图像帧中位于动态点云范围内的特征点剔除后再更新相机位姿，过程如图 4 中的图 4(a)，而在图 4(b)中，绿色框所判断的动态物体给相机图像的动态特征点(红色框内的点)提供判断范围和依据：



(a) 点云帧与相机图像融合流程　　(b) 点云簇动态物体确认和动态特征点判断

Figure 4. Diagram of fusion flow chart and schematic
图 4. 融合流程图和融合示意图

算法流程见表 4：

Table 4. Fusion algorithm
表 4. 融合算法

算法 4.融合算法
输入：相机前端特征点集合 $S=(kp_1,kp_2,\cdots,kp_n)$；激光雷达点云簇 $C_{ij}=(C_1,C_2,\cdots,C_k)$，$i,j$ 代表该物体被相邻两帧标记的序号。
输出：机器人融合位姿 T^*

```
1)for each points cluster C in C_ij
2)judge whether C is dynamic object
3) if C is dynamic object
4)        for each key point in S
5)                if kp_i in C
6)                        eliminate kp_i in S
7)                endif
8)        endfor
9)    endif
10)endfor
11)UpdatePose by new key points S
```

4.3. 相机位姿的估算

双目相机的 SLAM 系统分为前端、后端、回环检测和建图四个部分，其中前端部分也称为视觉里程计。视觉里程计根据相邻图像的信息估计出粗略的相机运动，为后端提供较好的初始值。视觉里程计主要算法分为特征点法和直接法，特征点法因其稳定和对光照变化不敏感而成为主流方法。本文采用提取 ORB 特征点后经过上述方法去掉动态物体特征点，解除动态物体对结果的干扰，最终采用 Bundle Adjustment (光束平差法)求解相机的位姿 $T=[R\,|\,t]$。

假设空间点世界坐标 $P_i=\left[X_i,Y_i,Z_i\right]^{\mathrm{T}}$，与其对应的图像像素坐标为 $\boldsymbol{u}_i=\left[u_i,v_i\right]^{\mathrm{T}}$，则他们之间的映射关系可以表示为：

$$s_i\boldsymbol{u}_i=KTP_i=K\left[R\,|\,t\right]P_i+W_i \tag{22}$$

$$W_i=s_i\boldsymbol{u}_i-K\left[R\,|\,t\right]P_i \tag{23}$$

在上式中，旋转矩阵 R 和 t 是未知项，而观测值 $\boldsymbol{u}_i$ 和真实的 3D 点 P_i 之间也存在着误差 W_i，对所有的误差求和，即可构建出最小二乘问题并将求解最佳相机位姿 T^* 转换为对误差增量方程的最优化问题：

$$T^*=\arg\min_T\frac{1}{2}\sum_{i=1}^{n}\left\|\boldsymbol{u}_i-\frac{1}{s_i}KTP_i\right\|_2^2 \tag{24}$$

为了使用高斯牛顿法，先将观测方程进行一阶泰勒展开，其中，$J\left(x\right)^{\mathrm{T}}$ 为 $e\left(x\right)$ 关于 x 的雅可比矩阵：

$$e\left(x+\Delta x\right)\approx e\left(x\right)+J\left(x\right)^{\mathrm{T}}\Delta x \tag{25}$$

让优化目标变为 Δx^*，即，

$$\Delta x^*=\arg\min_{\Delta x}\frac{1}{2}\left\|e\left(x\right)+J\left(x\right)^{\mathrm{T}}\Delta x\right\|^2 \tag{26}$$

根据极值条件，对 Δx 求导，并令其为零，得到增量方程：

$$J\left(x\right)e\left(x\right)+J\left(x\right)J\left(x\right)^{\mathrm{T}}\Delta x=0 \tag{27}$$

其中 e 为像素坐标误差，Δx 表示的是旋转和平移的变化量，求解出 $J\left(x\right)^{\mathrm{T}}$ 成为求解相机位姿 T^* 的关键，所以将(25)中的变量换成李代数并使用 BCH (Baker-Campbell-Hausdorff)公式求解：

$$\frac{\partial e}{\partial\delta\xi}=\lim_{\delta\xi\to 0}\frac{e\left(\delta\xi\oplus\xi\right)-e\left(\xi\right)}{\delta\xi}=\frac{\partial e}{\partial P'}\frac{\partial P'}{\partial\delta\xi} \tag{28}$$

$\oplus$ 表示李代数上的左乘扰动，P' 为相机坐标系下空间点的坐标 $\left[X',Y',Z'\right]^{\mathrm{T}}$，与之对应的像素坐标关系为：

$$\begin{bmatrix}su\\sv\\s\end{bmatrix}=\begin{bmatrix}f_x&0&c_x\\0&f_y&c_y\\0&0&1\end{bmatrix}\begin{bmatrix}X'\\Y'\\Z'\end{bmatrix} \tag{29}$$

$$\begin{cases}u=f_x\dfrac{X'}{Z'}+c_x\\v=f_y\dfrac{Y'}{Z'}+c_y\end{cases} \tag{30}$$

结合(29)、(30)，可将(28)式推导为：

$$\frac{\partial e}{\partial P'}=-\begin{bmatrix}\dfrac{\partial u}{\partial X'}&\dfrac{\partial u}{\partial Y'}&\dfrac{\partial u}{\partial Z'}\\\dfrac{\partial v}{\partial X'}&\dfrac{\partial v}{\partial Y'}&\dfrac{\partial v}{\partial Z'}\end{bmatrix}=-\begin{bmatrix}\dfrac{f_x}{Z'}&0&\dfrac{f_xX'}{Z'^2}\\0&\dfrac{f_y}{Z'}&\dfrac{f_yY'}{Z'^2}\end{bmatrix} \tag{31}$$

$$\frac{\partial P'}{\partial\delta\xi}=\frac{\partial\left(TP\right)}{\partial\delta\xi}=\left(TP\right)^{\odot}=\begin{bmatrix}I&-P'^{\wedge}\\\mathbf{0}^{\mathrm{T}}&\mathbf{0}^{\mathrm{T}}\end{bmatrix} \tag{32}$$

将(31)式与(32)式的前三维 $\begin{bmatrix}I&-P'^{\wedge}\end{bmatrix}$ 相乘，得到重投影误差关于相机位姿李代数的雅可比矩阵[11]，

$$\frac{\partial e}{\partial \delta\xi} = -\begin{bmatrix} \frac{f_x}{Z'} & 0 & \frac{f_x X'}{Z'^2} & -\frac{f_x X'Y'}{Z'^2} & f_x + \frac{f_x X'^2}{Z'^2} & \frac{f_x Y'}{Z'} \\ 0 & \frac{f_y}{Z'} & \frac{f_y Y'}{Z'^2} & -f_y - \frac{f_y Y'^2}{Z'^2} & \frac{f_y X'Y'}{Z'^2} & \frac{f_y X'}{Z'} \end{bmatrix} \tag{33}$$

根据公式(33)可求出 J^{T}，然后根据增量方程求解 Δx，那么 $\Delta x = \delta\xi$，然后使用李代数可得到相机位姿 $T = \exp\left(\delta\xi^{\wedge}\right)\cdot T$。

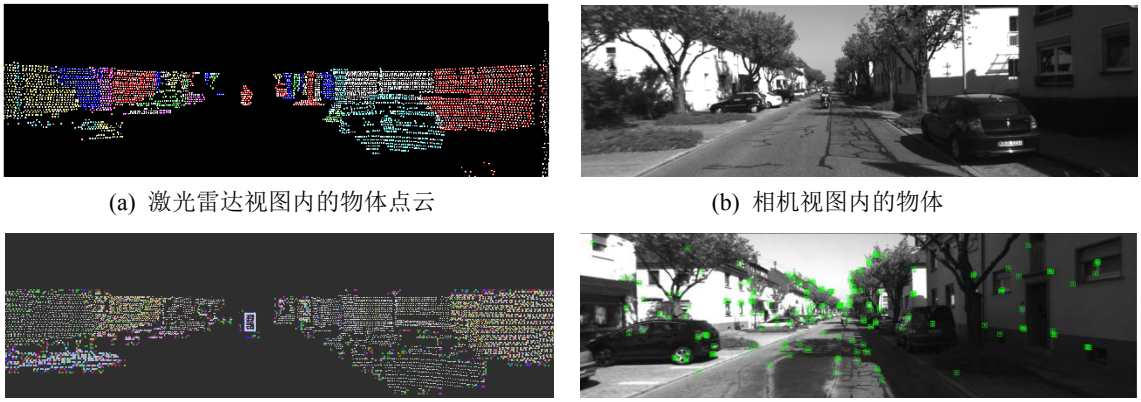
5. 实验结果与分析

5.1. 实验环境和数据集

本系统运行环境为 Intel(R) Xeon(R) CPU E5-26902.90GHz，16GB 内存台式机。本实验使用公开数据集 KITTI 提供的视觉图像和激光雷达数据。KITTI 数据集提供了 0~3 号共 2 组双目相机的视觉图像，其中 0 和 1 号相机为灰度相机，3 和 4 号相机为彩色相机；激光雷达为 64 线 3D 激光雷达 Velodyne HDL-64E × 1 (10 Hz, 64 laser beams, range: 100 m)；KITTI 在 yaml 文件中提供了各个传感器的安装参数；数据集场景中包含了行驶中的摩托车和运动的行人等，符合本实验需求场景。本文采用著名的视觉 SLAM 系统 ORB-SLAM2 [12] [13]加入本文的融合算法后所得结果跟原 ORB-SLAM2 纯视觉系统和纯激光系统 A-LOAM 进行综合的对比和评测。

5.2. 实验结果与分析

图 5 为相同场景下图 5(a)表示的将激光雷达点云聚类后分割出物体空间位置并将其显示到相机视图中的结果，本文用不同颜色对不同物体进行区分；图 5(b)是相机视图内物体的位置情况；图 5(c)采用 DBSCAN 算法聚类后所得的物体，经过动态物体的判断，发现了视图中的动态物体和动态物体的位置；图 5(d)中为前端提取到的特征点分布情况(绿色标记为特征点)，可以发现有部分特征点分布于动态场景中的动态物体上，本文融合算法在剔除了分布于已检出的动态物体上的特征点后，再转入后端优化。



(a) 激光雷达视图内的物体点云　(b) 相机视图内的物体

(c) 激光雷达视图中动态物体判断结果　(d) 相机前端提取特征点分布

Figure 5. The object point cloud in the lidar view and the object and feature point distribution in the camera view
图 5. 激光雷达视图中的物体点云和相机视图内的物体以及特征点分布情况

本文采用绝对轨迹误差(Absolute Trajectory Error, ATE)和相对位姿误差(Relative Pose Error, RPE)来整体评估算法的整体性能[14]。其中，绝对轨迹误差的计算方法为：

$$F_i := Q_i^{-1} S P_i \tag{34}$$

P_i 表示待估计的轨迹，Q_i 表示真实轨迹，i表示i时刻的误差，而对所有帧的均方根误差(RMSE)和方差(SSE)则可计算为：

$$\mathrm{RMSE}\left(F_{1:n}\right) := \left(\sum_{i=1}^{n}\left\|\mathrm{trans}\left(F_i\right)\right\|^2\right)^{1/2} \tag{35}$$

$$\mathrm{SSE}\left(F_{1:n}\right) := \sum_{i=1}^{n}\left\|\mathrm{trans}\left(F_i\right)\right\|^2 \tag{36}$$

相对位姿误差的计算法方法为：

$$E_i := \left(Q_i^{-1}Q_{i+\Delta}\right)^{-1}\left(P_i^{-1}P_{i+\Delta}\right) \tag{37}$$

Δ为前后两帧的时间差，则整个过程的均方根误差和方差为：

$$\mathrm{RMSE}\left(E_{i:n},\Delta\right) := \left(\sum_{i=1}^{n}\left\|\mathrm{trans}\left(E_i\right)\right\|^2\right)^{1/2} \tag{38}$$

$$\mathrm{SSE}\left(E_{i:n},\Delta\right) := \sum_{i=1}^{n}\left\|\mathrm{trans}\left(E_i\right)\right\|^2 \tag{39}$$

本文算法与 ORB-SLAM2、A-LOAM 算法计算后的绝对轨迹误差和相对位姿误差如表 5、表 6 所示。本文采用融合后的算法在对动态物体上提取到的特征点进行剔除后，有效降低了动态物体特征点对机器人位姿估算的影响，定位结果表现在轨迹误差和位姿误差上相比于 ORB-SLAM2 纯视觉系统降低了 2% 和 3‰；同样纯激光 A-LOAM 算法因为在没有去除动态物体激光点影响和激光点的稀疏性导致轨迹误差和位姿误差均高于本文算法。

Table 5. Comparison of absolute position error of algorithms
表 5. 算法绝对轨迹误差对比

算法	Max	Min	Median	Mean	RMSE	SSE
Merged	3.51351	0.08937	1.05532	1.13675	1.26489	7265.46068
ORB-SLAM2	3.44411	0.14402	1.06515	1.15297	1.27389	7369.12126
A-LOAM	10.0332	0.27378	3.15356	3.55423	4.12474	77258.5215

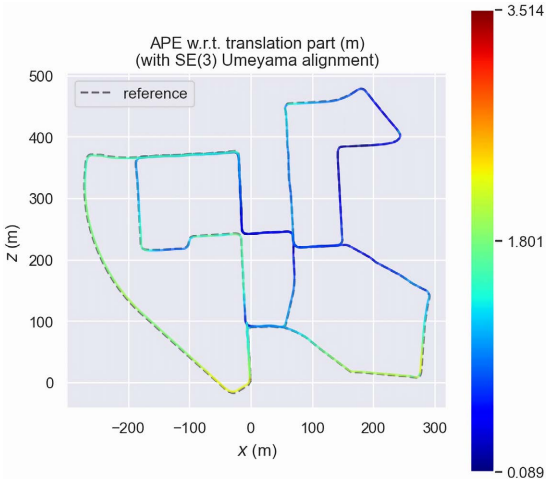
Table 6. Comparison of relative pose errors of algorithms
表 6. 算法相对位姿误差对比

算法	Max	Min	Median	Mean	RMSE	SSE
Merged	0.29974	0.00060	0.01498	0.01936	0.02808	3.58077
ORB-SLAM2	0.27703	0.00037	0.01516	0.01960	0.02822	3.61761
A-LOAM	1.19197	0.00654	1.20017	1.15572	1.21666	6720.48

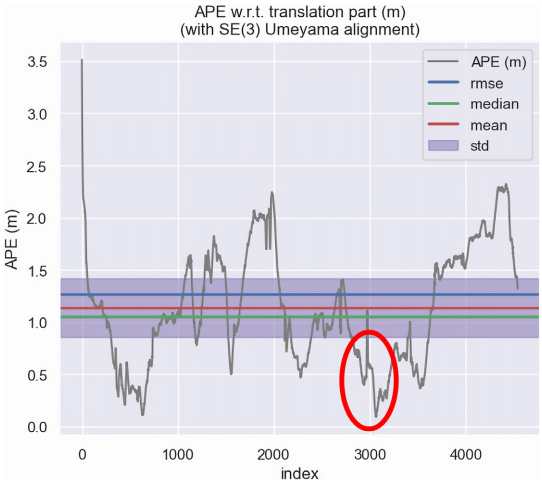
图 6 为融合算法与 ORB-SLAM2 的绝对轨迹误差和相对位姿误差的指标分布情况。从图 6(b)与图 6(f)和图 6(d)与图 6(h)的对比中发现，融合算法在滤除掉周围动态物体的影响后，轨迹更为平滑，位姿更为平稳。

图 7 为采用融合算法和视觉 ORB-SLAM2 在旋转、平移上的误差与运行里程和运行速度的关系。从图 7(a)和图 7(b)中可以看出，融合算法在距离越长的运行里程中相较于纯视觉算法较好地降低了旋转和平移误差；从图 7(c)中可以发现，融合算法和纯视觉算法在中速(40 公里每小时)以下旋转误差得到较快

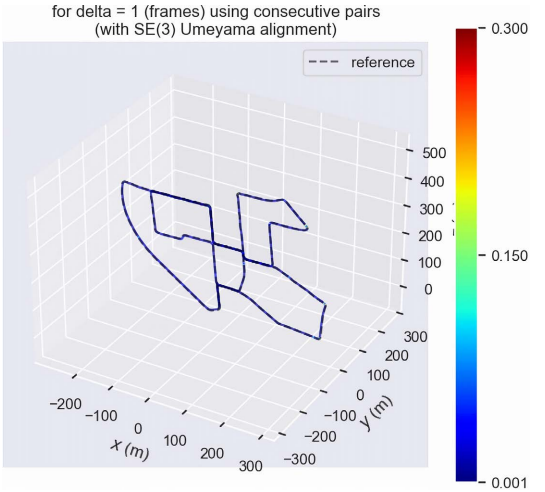
收敛，并且融合算法的旋转误差小于纯视觉算法；从图 7(d)的对比结果中可以发现，中低速下融合算法的平移误差优于纯视觉算法。



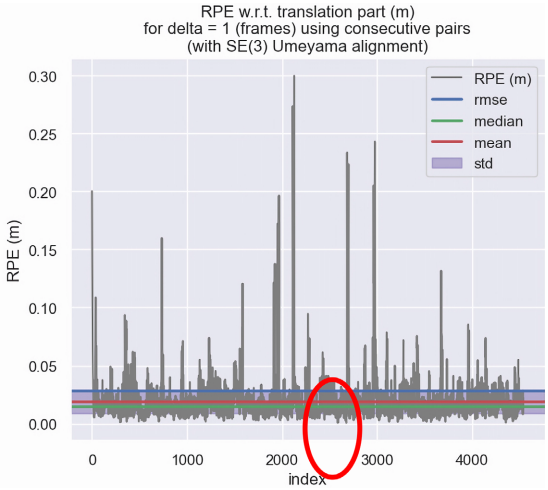
(a) Merged APE Map



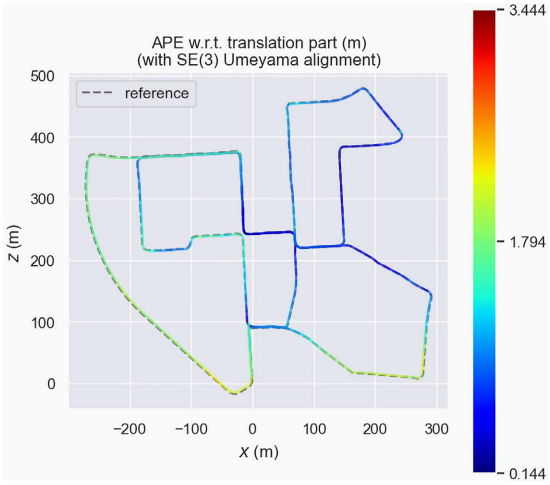
(b) Merged APE Raw



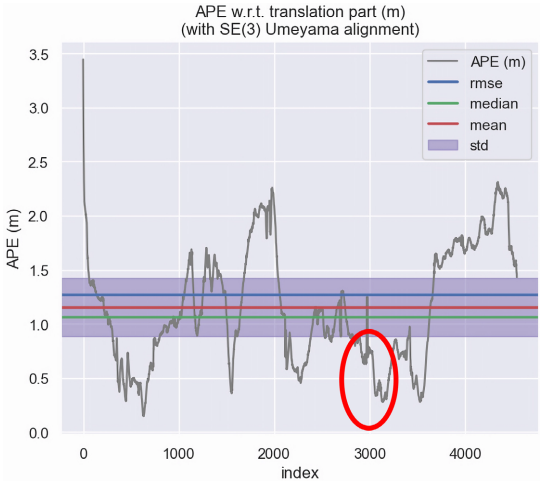
(c) Merged RPE Map



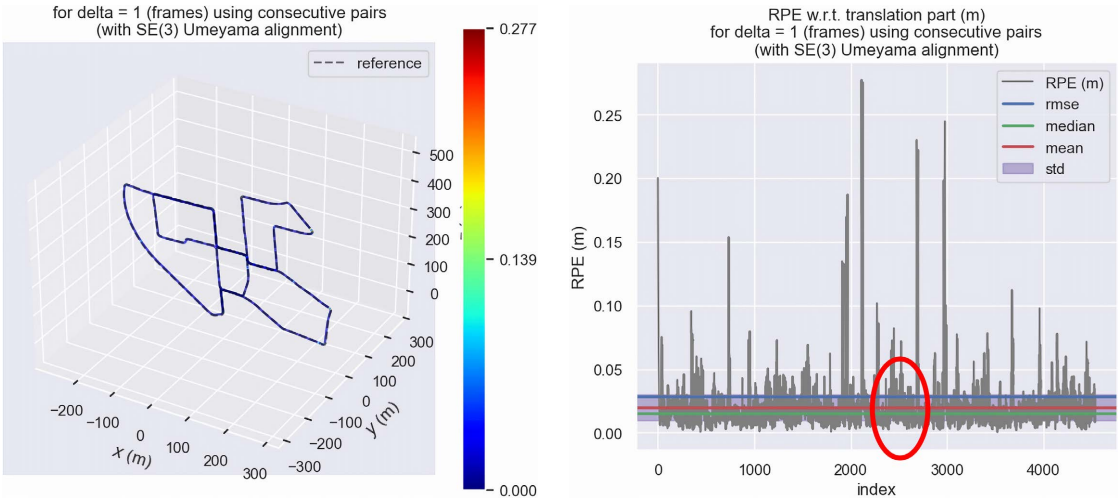
(d) Merged RPE Raw



(e) ORB-SLAM2 APE Map

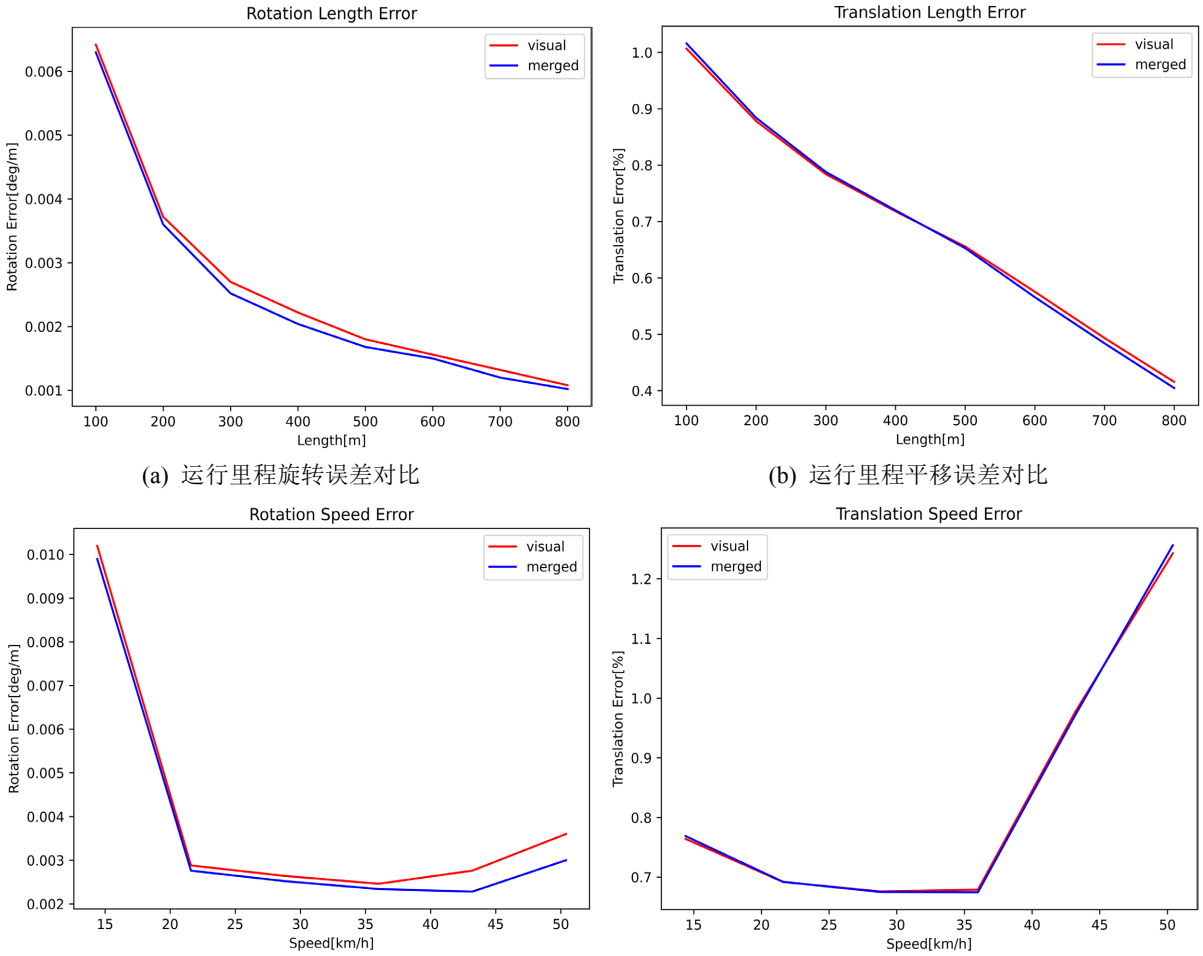


(f) ORB-SLAM2 APE Raw



(g) ORB-SLAM2 RPE Map (h) ORB-SLAM2 RPE Raw

Figure 6. Absolute trajectory error and relative pose error distribution of the two algorithms
图 6. 两种算法的绝对轨迹误差与相对位姿误差分布



(a) 运行里程旋转误差对比 (b) 运行里程平移误差对比

(c) 运行速度旋转误差对比 (d) 运行速度平移误差对比

Figure 7. Comparison of fusion algorithm and ORB-SLAM2 algorithm on rotation and translation
图 7. 融合算法与 ORB-SLAM2 算法关于旋转、平移的对比

6. 结论

本文针对环境中存在的动态物体影响 SLAM 位姿估算的准确性，提出了一种实现激光雷达数据和相机数据的融合算法。算法首先通过对激光雷达点云使用 DBSCAN 算法进行聚类，然后对聚类点云簇采用改进的评分和匈牙利匹配实现相邻帧点云簇的匹配，并通过判断相邻帧点云簇的运动性确定运动点云的相机坐标系坐标，最后剔除运动点云簇内的视觉特征点再进行相机位姿估算，达到提高视觉里程计定位精度的效果。该方法的优点在于，一是使用的点云非监督聚类算法不需要预先设置环境中的物体个数，符合实际使用场景中机器人对周围环境未知的情况；二是采用的相邻帧物体匹配方法使用的是稀疏的点云簇，不需要对物体类别或者颜色进行额外判断，避免了大量的运算开支。在 KITTI 数据集的验证中可以看出，本算法可滤除动态特征点的影响，有效提高里程计的定位精度。但同时也发现本方法在机器人高速运动的情况下对动态物体的判断出现了一些误差导致了结果的微小发散，在以后的工作中，将进一步研究对动态物体判断的更有效方法，全面提升系统的精确性。

参考文献

[1] Yu, C., Liu, Z., Liun X., *et al*. (2018) DS-SLAM: A Semantic Visual SLAM towards Dynamic Environments. *Proceedings of* 2018 *IEEE/RSJ International Conference on Intelligent Robots and Systems* (*IROS*), Madrid, Spain, 01-05 October 2018. https://doi.org/10.1109/IROS.2018.8593691

[2] Chen, X., Li, S., Mersch, B., *et al*. (2021) Moving Object Segmentation in 3D LiDAR Data: A Learning-Based Approach Exploiting Sequential Data. *IEEE Robotics and Automation Letters*, **6**, 6529-6536. https://doi.org/10.1109/LRA.2021.3093567

[3] Ester, M., Kriegel, H.-P., Sander, J. and Xu, X.W. (1996) A Density-Based Algorithm for Discovering Clusters in Large Spatial Databases with Noise. *Proceedings of the Second International Conference on Knowledge Discovery and Data Mining*, Portland, Oregon, August 1996, 226-231.

[4] 杨慧林. 基于四线激光雷达的目标检测与跟踪算法研究[D]: [硕士学位论文]. 长沙: 湖南大学, 2018.

[5] 孙浩博, 杨开明, 朱煜, 鲁森. 基于密度聚类的磁悬浮平面电机模态参数估计[J]. 清华大学学报: 自然科学版, 2023, 63(1): 33-43.

[6] 高翔. 视觉 SLAM 十四讲: 从理论到实践[M]. 北京: 电子工业出版社, 2017.

[7] 徐威扬. 跨指令集架构二进制漏洞搜索技术研究[D]: [硕士学位论文]. 长沙: 国防科学技术大学, 2017.

[8] 袁夏. 基于激光雷达的智能机器人环境理解关键技术研究[D]: [博士学位论文]. 南京: 南京理工大学, 2010.

[9] 张恒, 徐万红, 张禹. 激光雷达与深度相机融合的 SLAM 技术研究[J]. 机械工程师, 2020(7): 22-29.

[10] 张俐, 郭超朋, 刘京亮. CMM 测量中基于确定采样策略的圆锥面拟合方法[J]. 测试技术学报, 2018, 32(1): 1-6.

[11] 徐陈, 周怡君, 罗晨. 动态场景下基于光流和实例分割的视觉 SLAM 方法[J]. 光学学报, 2022, 42(14): 139-151.

[12] Mur-Artal, R., Montiel, J.M.M. and Tardos, J.D. (2015) ORB-SLAM: A Versatile and Accurate Monocular SLAM System. *IEEE Transactions on Robotics*, **31**, 1147-1163. https://doi.org/10.1109/TRO.2015.2463671

[13] Mur-Artal, R. and Tardos, J.D. (2017) ORB-SLAM2: an Open-Source SLAM System for Monocular, Stereo and RGB-D Cameras. *IEEE Transactions on Robotics*, **33**, 1255-1262. https://doi.org/10.1109/TRO.2017.2705103

[14] 艾青林, 王威, 刘刚江. 室内动态环境下基于网格分割与双地图耦合的 RGB-D SLAM 算法[J]. 机器人, 2022, 44(4): 431-442.