Faster-LIO 复现研究报告

摘要

Faster-LIO 作为一种轻量级、高效的激光雷达-惯性里程计项目，满足了自动驾驶、机器人导航等领域对高精度、实时性定位与建图技术的需求。Faster-LIO 通过算法优化，实现了更高的算法效率，提高了系统的实时性和响应速度。其应用场景广泛，包括自动驾驶汽车的路径规划、机器人导航与避障等。复现 Faster-LIO 是深入理解其算法原理、验证性能、探索应用潜力及促进学术交流的重要途径。本文成功复现了 Faster-LIO 项目，并通过离线测试和在线测试验证了其性能。测试结果表明 Faster-LIO 能准确跟踪运动轨迹并输出稳定里程计信息，在减少局部轨迹漂移和保持全局轨迹一致性方面具有优势。

关键词：Faster-LIO ；激光雷达-惯性里程计；稀疏增量体素；无人机

1 引言

随着自动驾驶、机器人导航等领域的快速发展，定位与建图技术成为这些领域中的核心技术之一。传统的定位与建图方法在面对复杂多变的环境时，往往存在精度不足、实时性差等问题。Faster-LIO 作为一种轻量级、高效的激光雷达-惯性里程计项目，其出现满足了这些领域对高精度、实时性定位与建图技术的迫切需求。Faster-LIO 在算法上进行了优化，通过替换增量式 k-d 树为增量式稀疏体素结构（iVox），实现了更高的算法效率。这种优化使得 Faster-LIO 能够在不同类型的激光雷达上实现更高的帧率，从而提高了系统的实时性和响应速度。这对于自动驾驶和机器人导航等需要快速决策的应用场景来说至关重要。Faster-LIO 的应用场景非常广泛，包括但不限于自动驾驶汽车的路径规划、机器人导航与避障、无人机巡检与地形测绘等。这些应用场景对定位与建图技术的要求各不相同，但都对精度和实时性有着极高的要求。Faster-LIO 的出现为这些应用场景提供了更加可靠和高效的解决方案。

复现 Faster-LIO 的过程是一个深入理解其算法原理的过程。通过亲自实现 Faster-LIO，可以更深入地了解其内部的工作机制，包括数据预处理、特征提取、匹配与优化等关键步骤。这对于进一步改进和优化算法具有重要意义。同时可以验证其在不同场景下的性能表现，对于评估 Faster-LIO 在实际应用中的可行性具有重要

意义。最后，复现 Faster-LIO 也是促进学术交流与合作的重要途径。通过分享复现经验和成果，可以与同行进行深入的交流和讨论，共同推动定位与建图技术的发展和进步。

综上所述，Faster-LIO 的出现满足了自动驾驶、机器人导航等领域对高精度、实时性定位与建图技术的需求，而复现 Faster-LIO 则是深入理解其算法原理、验证其性能、探索其应用潜力以及促进学术交流与合作的重要途径。

2 相关工作

2.1 激光雷达-惯性里程计算法

激光雷达-惯性里程计算法（LiDAR-Inertial Odometry，简称 LIO）是一种融合激光雷达（LiDAR）和惯性测量单元（IMU）数据以实现精准定位和导航的算法。LIO 算法结合了激光雷达和 IMU 的互补优势。激光雷达通过发射激光束并测量反射光线的时间或相位差来确定物体与传感器的距离，从而生成周围环境的 3D 点云数据。IMU 则提供关于设备运动（如加速度和角速度）的实时信息。LIO 算法将这两种数据进行融合，通过迭代优化算法（如扩展卡尔曼滤波器或非线性优化方法）来估计设备的位姿（位置和姿态）。

2.2 Fast-LIO

Fast-LIO（Fast LiDAR-Inertial Odometry）通过紧耦合的迭代扩展卡尔曼滤波器（EKF）融合激光雷达特征点和 IMU 数据，实现了在快速运动、噪声或杂乱环境中的稳健导航。大致流程是将激光雷达输入到特征提取模块，可获得平面和边缘特征，将提取后的特征和 IMU 测量值进行状态估计，输出 10HZ 到 50HZ 的状态估计结果，并根据状态估计结果将特征点注册到全局地图。Fast-LIO 的核心技术包括：

（1）快速迭代卡尔曼滤波：用于里程计优化，确保实时性和准确性。

（2）自动初始化：在大多数稳定环境中自动启动，简化了使用流程。

（3）并行 KD-Tree 搜索：减少计算负担，提高处理速度。

2.3 Fast-LIO2.0

对于 Fast-LIO2 而言，相较于 Fast-LIO，做了如下改进：

(1)不用线、面特征点而使用全局点云,通过原始点云与地图的配准，可以有效地利用环境中的细微特征，从而提高准确性，同时不使用特征提取也可以更好地适应

不同的激光雷达。

(2) 使用 ikd-tree 存储点云,以高效地表示大型稠密点云地图,除了高效的最近邻搜索外，新的数据结构还支持增量地图更新（即点插入、下采样、点删除）和以最小计算成本进行动态平衡

2.4 Faster-LIO

Faster-LIO 作为Fast-LIO2 的升级版,通过一些优化措施进一步提升了处理速度。该算法不再采用复杂的基于树的数据结构来组织空间点云，而是采用了一种名为增量体素(iVox)的点云空间数据结构，这种结构是从传统的体素演变而来，支持点云的增量插入和并行近似 k-NN 查询。使用增量体素(iVox)存储点云，与 Fast-LIO2 相比的实验结果：精度方面两者相当，但 Faster-LIO 通过牺牲一小部分配准精度来减少处理时间。在大多数数据集上，处理时间减少了约一半。

3 本文方法

3.1 算法流程

（1）数据采集

激光雷达数据：激光雷达持续扫描周围环境，生成点云数据，这些数据提供了环境的三维结构信息。

惯性测量单元（IMU）数据：IMU 提供关于设备当前加速度和角速度的数据，帮助估计设备的运动状态和姿态变化。

（2）预处理

点云去畸变：由于激光雷达在移动过程中采集数据，移动引起的畸变需要被校正。使用 IMU 数据来估计扫描期间的运动，对点云数据进行去畸变处理。

IMU 数据预积分：将连续的 IMU 数据进行预积分，为后续的状态估计提供粗略的运动估计。

（3）状态估计

初始化：使用 IMU 数据初始化系统的姿态、速度和位置。

运动估计：结合 IMU 的预积分结果和点云数据，使用非线性优化方法（如图优化）来估计设备的运动轨迹和姿态。这一步通常涉及到最小化重投影误差，即点云数据与预测模型之间的差异。

（4）地图构建

局部地图更新：根据新的点云数据更新局部地图。局部地图用于快速匹配和位置估计。

全局地图维护：将局部地图合并到全局地图中，全局地图用于长期导航和路径规划。

（5）回环检测与优化

回环检测：检测是否返回到先前访问过的位置。如果检测到回环，即当前的点云与历史点云高度相似，这表明位置的重合。

图优化：在检测到回环时，通过图优化技术调整整个地图，减少累积误差，优化轨迹和地图的一致性。

（6）状态更新与输出

状态更新：根据估计的运动和位置更新系统状态。

输出：输出当前的位置、姿态和构建的地图数据，供导航和其他应用使用。

Faster-LIO 通过这些步骤实现了高效的实时定位和地图构建，特别适用于动态环境和高速移动的场景。这种方法的关键在于有效地融合激光雷达和 IMU 数据，以及高效的算法实现，以保证实时性和准确性。

3.2 主要特点

（1）高精度：通过紧耦合的激光雷达和惯性测量单元数据，Faster-LIO 能够实现高精度的姿态估计和跟踪。

（2）高实时性：利用并行稀疏增量体素技术，Faster-LIO 能够加速点云地图的构建过程，适用于实时应用场景。在固态激光雷达上，Faster-LIO 可以实现约 1k-2k Hz 的帧率；在典型的 32 线旋转激光雷达上，也可以实现超过 100 Hz 的帧率。

（3）灵活性：Faster-LIO 支持多种激光雷达类型，包括固态激光雷达和旋转激光雷达等。同时，它还支持离线模式和在线模式，用户可以根据需求选择合适的运行方式。

（4）易扩展性：Faster-LIO 的代码结构清晰，易于与其他算法或传感器数据进行集成和扩展。

4 复现细节

4.1 环境搭建

（1）操作系统与 ROS 版本

操作系统：Ubuntu 20.04

ROS 版本：Noetic

（2）依赖安装

sudo apt-get install libgoogle-glog-dev

sudo apt-get install libeigen3-dev

sudo apt-get install libpcl-dev

sudo apt-get install libyaml-cpp-dev

（3）编译器升级

由于 Faster-LIO 使用的标准为 C++17，因此需要升级 gcc 和 g++。

sudo add-apt-repository ppa:ubuntu-toolchain-r/test

sudo apt update

sudo apt install gcc-9 g++-9

（4）TBB 库安装

从 GitHub 下载并编译 TBB 库，然后安装到系统中。

（5）HDF5 库软连接

在指定目录下创建文件夹，并创建 HDF5 库的软连接。

4.2 源码下载与编译

（1）克隆 Faster-LIO 源码到 ROS 工作空间:

git clone https://github.com/gaoxiang12/faster-lio.git

（2）进入 build 目录，使用 cmake 编译项目:

cd ~/catkin_ws/src

cp -r ~/faster-lio .

cd ~/catkin_ws

catkin_make

（3）编译成功后，运行以下命令进行 ROS 设置:

source devel/setup.bash

4.3 离线测试

下载 rosbag 数据包（https://zenodo.org/records/10122133）；启动 ROS 核心：

打开一个新的终端窗口，输入 roscore 命令启动 ROS 核心；播放 rosbag 数据包：在另一个终端窗口中，使用 rosbag play 命令播放之前下载的 rosbag 数据包；运行 Faster-LIO 节点：在第三个终端窗口中，使用 roslaunch 命令启动 Faster-LIO 的映射或定位节点；开始处理 rosbag 数据包中的激光雷达和 IMU 数据，并输出里程计信息或其他定位结果。如下图所示公开数据使用 faster-lio 算法运行成功：

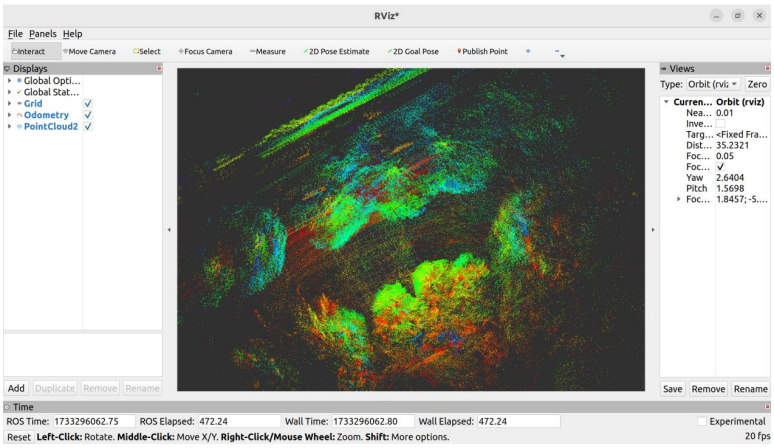

图 1 公开数据集测试成果

4.4 在线测试与精度分析

无人机搭载 Lidar+IMU+RTK+电脑运行 faster-lio 采集汇星楼二楼的点云数据。无人机采集获得轨迹数据及点云数据。 Faster-LIO 轨迹和 GNSS 轨迹对比如图 2 所示，以 GNSS 轨迹为真值来计算 Faster-LIO 轨迹与 GNSS 轨迹的欧氏距离均方根误差(RSM)，以量化评价 Faster-LIO 的轨迹精度，其 RSM 为 0.17969 米，误差较小，可见 Faster-LIO 在减少局部轨迹漂移和保持全局轨迹一致性方面具有显著优势。

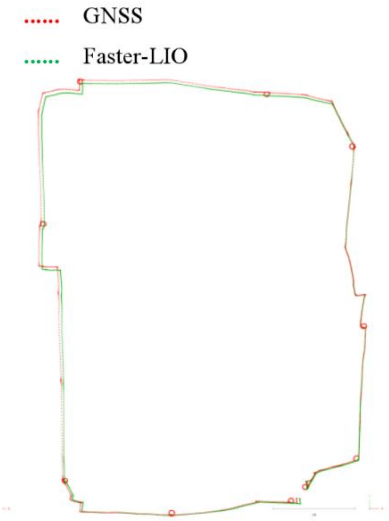

图 2 Faster-LIO 轨迹和 GNSS 轨迹对比

5 总结与展望

本文成功复现了 Faster-LIO 项目，并对其算法原理和性能进行了深入理解和验证。离线测试部分，使用 rosbag 数据包作为输入数据，成功运行了 Faster-LIO 的映射和定位节点。测试结果显示，Faster-LIO 能够准确地跟踪激光雷达和 IMU 的运动轨迹，并输出稳定的里程计信息。此外，还尝试了在线测试，使用无人机搭载 Lidar+IMU+RTK+电脑系统进行了数据采集和测试。测试结果表明，Faster-LIO 在减少局部轨迹漂移和保持全局轨迹一致性方面具有显著优势，其轨迹精度与 GNSS 轨迹相比误差较小。未来的研究中可以继续通过在实际场景中进行测试和应用可以发现 Faster-LIO 的优势和不足，并据此进行针对性的改进和优化，还可以结合其他传感器或算法，进一步拓展 Faster-LIO 的应用范围和功能。

参考文献

[1]C.Bai, T. Xiao, Y. Chen, H. Wang, F. Zhang and X. Gao. Faster-LIO: Lightweight Tightly Coupled Lidar-Inertial Odometry Using Parallel Sparse Incremental Voxels. IEEE Robotics and Automation Letters, 7(2):4861-4868, 2022.