R^{2}DIO: A Robust and Real-Time Depth-Inertial Odometry Leveraging Multimodal Constraints for Challenging Environments

Jie Xu, Ruifeng Li, Song Huang, Xiongwei Zhao, Shuxin Qiu, Zhijun Chen, and Lijun Zhao, *Member, IEEE*

***Abstract*— RGB-D cameras serve as indispensable sensors for indoor simultaneous localization and mapping (SLAM) in lightweight robots. However, many RGB-D SLAM systems fail to capitalize on the multimodal information provided by cameras due to computational constraints, leading to suboptimal performance in challenging environments such as structure-less scenes for LiDARs and texture-less scenes for cameras. To address this issue, we propose a novel, lightweight, and robust real-time depth-inertial odometry (R^{2}DIO) designed for time-of-flight (ToF) RGB-D cameras. It effectively extracts pseudo 3-D line and plane features from color and depth images through the utilization of agglomerative hierarchical clustering (AHC), which leverages the adjacency relationships between pixels and incorporates multimodal constraints. To enhance real-time performance, directional consistency constraints are applied to filter mismatches during feature alignment. R^{2}DIO estimates states and generates dense colored maps using line and plane matching constraints, inertial measurement unit (IMU) preintegration constraints, and historical odometry constraints. Experimental results underscore the robustness, accuracy, and efficiency of R^{2}DIO. It can accurately locate in structure-less or texture-less scenes and operate at 30 Hz on a low-power platform. We publicly provide R^{2}DIO's source code and experiment datasets to foster community development.**

***Index Terms*— Dense reconstruction, direction consistency (DC) constraint, indoor simultaneous localization and mapping (SLAM), inertial measurement unit (IMU), multimodal, plane, pseudo 3-D line, RGB-D, time of flight (ToF).**

Manuscript received 25 July 2023; revised 7 September 2023; accepted 15 September 2023. Date of publication 29 September 2023; date of current version 27 October 2023. This work was supported in part by the National Natural Science Foundation of China under Grant 62073101 and in part by the Self-Planned Task of State Key Laboratory of Robotics and System (HIT) under Grant SKLRS202007B. The Associate Editor coordinating the review process was Dr. Alvaro Hernandez. *(Corresponding author: Lijun Zhao.)*

Jie Xu and Lijun Zhao are with the State Key Laboratory of Robotics and Systems, Harbin Institute of Technology, Harbin 150001, China, and also with the Yangtze River Delta HIT Robot Technology Research Institute, Wuhu 241000, China (e-mail: jeff_xu_0503@foxmail.com; zhaolj@hit.edu.cn).

Ruifeng Li is with the State Key Laboratory of Robotics and Systems, Harbin Institute of Technology, Harbin 150001, China (e-mail: lrf100@hit.edu.cn).

Song Huang, Shuxin Qiu, and Zhijun Chen are with the Yangtze River Delta HIT Robot Technology Research Institute, Wuhu 241000, China (e-mail: mmerit31@163.com; qsx97@foxmail.com; 1299771369@qq.com).

Xiongwei Zhao is with the School of Electronic and Information Engineering, Harbin Institute of Technology (Shenzhen), Shenzhen 518071, China (e-mail: xwzhao@stu.hit.edu.cn).

Data is available on-line at https://github.com/jiejie567/R2DIO and https://www.youtube.com/watch?v=YqJZTE948sk.

Digital Object Identifier 10.1109/TIM.2023.3320753

NOMENCLATURE

A. Expression

$\mathrm{Exp}(\cdot)/\mathrm{Log}(\cdot)$	Rodrigues' transformation between the rotation matrix and rotation vector.
$^{G}(\cdot)$	Value of $(\cdot)$ expressed in global frame.
$^{I}(\cdot)$	Value of $(\cdot)$ expressed in IMU frame.
$\times$	Cross product between vectors.
$\cdot$	Dot Product between vectors.
$\|\cdot\|$	Euclidean norm.

B. Variables

$\boldsymbol{\phi}$	Rotation in Lie Algebras $\mathfrak{so}(3)$.
$\mathbf{p}$	Position $\in \mathbb{R}^3$.
$({}^{G}\boldsymbol{\phi}_I, {}^{G}\mathbf{p}_I)$	IMU attitude and position with respect to global frame.
$\mathbf{b^a}$,$\mathbf{b^g}$	Bias of gyroscope and accelerometer in the IMU.
$\boldsymbol{G}$	Gravitational acceleration.
$\mathbf{v}$	Linear velocity.
$\mathbf{x}$	Accurate state.
$\hat{\mathbf{x}}$	Prior estimation of $\mathbf{x}$.

I. INTRODUCTION

RGB-D cameras play important roles in a wide range of indoor applications, such as navigation, dense reconstruction, and augmented/virtual reality. They can provide multimodal information such as RGB images, depth images, and inertial measurement unit (IMU) measurements, which bring diverse constraints to robot pose estimation.

Based on their ranging methods, RGB-D cameras can be divided into two types: time of flight (ToF) and structured light. Compared to structured light cameras, ToF cameras have higher ranging accuracy and a longer measurement range. They also have low power consumption and are small in size, making them suitable for small and lightweight indoor robots. However, ToF RGB-D cameras also have some disadvantages, such as a small field of view (FoV), short coverage range, and vulnerability to light interference in the depth module, which may cause simultaneous localization and mapping (SLAM) to fail. Therefore, it is essential to fuse multimodal information to improve the robustness and accuracy of SLAM for ToF RGB-D cameras. At the same time, it is also necessary to make the system lightweight to adapt to small indoor robots. ToF RGB-D cameras can accurately measure depth and capture the structural features of an environment. Additionally, their ability to instantaneously acquire all point clouds within a frame ensures that the generated point clouds remain free from motion distortion caused by movement.

Numerous RGB-D systems exist, yet many fail or experience significant drift in challenging environments. Challenging environments are characterized by a lack of texture for visual sensors and a scarcity of structural features for depth sensors, resulting in insufficient constraints and performance degradation. These system failures likely stem from the inadequate consideration of multimodal constraints and suboptimal real-time performance. Without the assistance of IMU prior pose, ORB-SLAM2 [1] readily loses track during rapid rotational movements with large angular changes. In contrast, ORB-SLAM3 [2] solely utilizes textural feature points, rather than structural features, causing it to fail in environments with abundant structural features but lacking texture. BundleFusion [3] exhibits exceptional reconstruction performance but necessitates a GPU for real-time capability. SSL-SLAM [4] is the only open-source system specifically designed for ToF RGB-D cameras. Due to the high accuracy of depth values, it employs the LOAM [5] method to construct the system framework. However, it extracts features exclusively from the depth image, and the RGB module serves only to color point clouds. This limitation leads to failure in environments with rich textures but scarce structural features, such as the expansive exhibition hall in Fig. 1(a).

The utilization of IMUs can be an optimal approach for overcoming the limitations of odometry in the short term that arise due to factors such as poor lighting conditions for cameras or environments with low structural complexity for LiDARs. In addition, the high-frequency kinematic measurements provided by IMUs can aid in correcting motion distortions in LiDAR scans, particularly when the robot is traveling at high speeds.

To address these issues and fuse information from multiple sensors, we propose R^{2}DIO: a robust, real-time, depth-inertial indoor SLAM system utilizing a ToF RGB-D camera and a built-in IMU. The system can construct a dense colored map, as shown in Fig. 1. We evaluate our system on real data, including slow and fast movements, low texture, and low structure. The main contributions of this article are as follows.

1) We propose a novel depth-inertial SLAM system designed for ToF RGB-D cameras. The system utilizes a factor graph as its foundation and incorporates multimodal constraints, including IMU measurements, color textural features, and depth structural features, instead of relying solely on a single modal information. Consequently, it can effectively work in textureless or structureless scenarios where other systems may fail. Furthermore, we have open-sourced our implementation and datasets on GitHub for the benefit of the community.
2) We efficiently utilize the adjacency of pixel data in color and depth maps by employing the agglomerative hierarchical clustering (AHC) method to extract pseudo 3-D line and plane features, as opposed to extracting from point clouds where adjacency is disrupted. Besides, the pseudo 3-D line includes text and structure constrains. This approach results in a twice increase in feature extraction speed.
3) We propose employing direction consistency (DC) constraints to expedite the alignment of features, increasing the processing speed from 10 to 30 Hz. This methodology involves assigning a direction vector to each individual feature point and leveraging the accuracy of nearest-neighbor matching, determined by these direction vectors, to minimize the number of iterations.
4) Our system's exceptional robustness and high efficiency in various challenging indoor scenes have been verified by comparing it to SSL-SLAM and other high-performing SLAM systems.

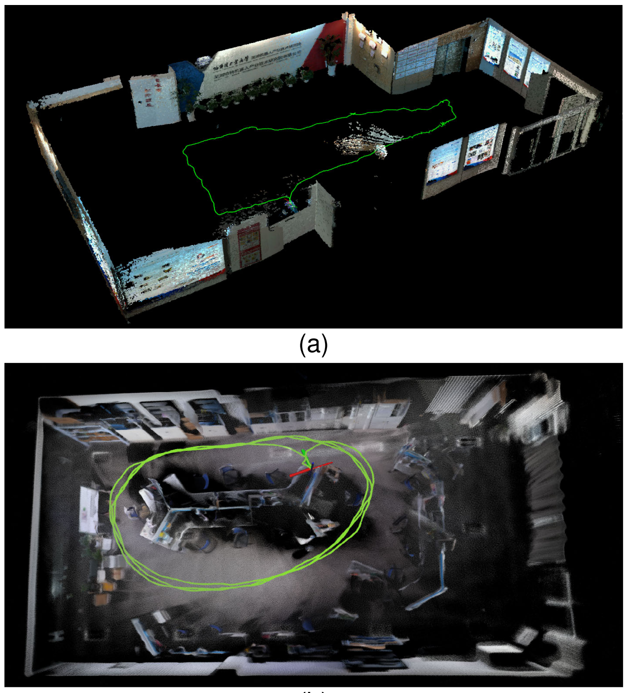


Fig. 1. Our system can generate a dense, real-time 3-D-colored point cloud of the indoor environment. The estimated trajectory of robot motion is depicted as a green path, with the start and end points aligned. (a) Map constructed with RealSense L515 in the exhibition hall, whereas (b) presents a top view map created with Azure Kinect DK by traversing three indoor circles. Notably, the text in (a) is clearly visible.

The remainder of this article is organized as follows. Section II reviews related works on existing LiDAR SLAM and RGB-D SLAM systems. Section III details the proposed system, including IMU state propagation and preintegration, feature extraction and alignment, ranging error model, state estimation, and mapping. Section IV presents experimental results in comparison with other SLAM systems, followed by a discussion in Section V and conclusion in Section VI.

II. Related Work

Our system for ToF RGB-D cameras incorporates insights from LiDAR SLAM; therefore, we review LiDAR and RGB-D SLAM in this section.

A. LiDAR SLAM

LiDAR offers high measurement accuracy, an extensive measuring range, and a wide FoV. However, due to the large number of points in one scan, methods like [6] that directly use iterative closest point (ICP) [7] for point cloud registration are often less real-time and sensitive to noisy point clouds. Ji and Singh [5] proposed LOAM, a system that aligns edge features and surf features in the feature space, rather than registering all points. LOAM consists of two threads, a scan-to-scan match thread, and a scan-to-map refinement thread (1 Hz) for real-time capability and precision, respectively. Regrettably, distortion compensation and state estimation are solved by iterative calculation, which is time-consuming. To address this, Wang et al. [8] proposed F-LOAM, which adopts a noniterative two-stage distortion compensation method, achieving excellent localization accuracy with a processing rate exceeding 10 Hz.

These works solve distortion compensation by assuming constant velocity robot motion, which is inapplicable in some scenes with abrupt movement changes. Consequently, some systems incorporate IMU measurements to infer motion and solve distortion compensation. LIO-SAM [9] estimates states based on a factor graph, including IMU preintegration to constrain the state between two frames. The IMU can also provide initial values for point cloud registration and correct point cloud distortion. Besides optimization-based approaches, FAST-LIO [10], [11] and LINS [12] are tightly-coupled LiDAR-inertial systems that estimate poses via iterated Kalman filtering, achieving high-precision and high-efficiency positioning. VoxelMap [13] conducts error analysis on probabilistic planes based on FAST-LIO, improving accuracy. These LiDAR SLAM algorithms exhibit high accuracy outdoors but may experience system drift indoors with narrow FoV LiDARs, like Livox Avia, in simple-structured environments.

B. RGB-D SLAM

Over the last decade, numerous systems have been developed for dense reconstruction and SLAM. KinetFusion [14] and ElasticFusion [15] focus on dense reconstruction, while ORB-SLAM2 [1] supports mono, stereo, and RGB-D cameras while offering real-time capability. SSL-SLAM [4], based on LOAM's feature extraction and alignment method, estimates robot poses using depth images and colors point clouds with RGB images. Compared to DSO [16], RGB-D DSO [17] directly obtains key points' depth values and uses photometric error to assist pose estimation in pixel areas without depth values. Some works [18], [19] exploit Manhattan World (MW) planes to estimate drift-free rotation. Although these systems perform well in experiments, they can easily fail because the MW assumption is only applicable in specific environments.

The aforementioned systems can suffer from degeneration in certain cases, including fast motion, dynamic scenarios, and weak textures. To address these issues, some systems integrate IMU and RGB-D information. For example, ORB-SLAM3 [2], based on ORB-SLAM2, tightly couples IMU measurements and uses a multimap module to achieve high accuracy and robustness. VINS-RGBD [20], [21] fuses depth information into VINS-Mono [22] to stabilize scale and accelerate system initialization. SSL-SLAM [23] uses an IMU to infer prior poses for accurate feature matching and employs IMU preintegration as a constraint between two frames. However, its real-time performance is poor (10 Hz), making it prone to failure in fast-motion scenarios.

III. Methodology

First, we define the frame notations used throughout this article. The initial IMU frame serves as the global frame G, and the IMU frame is denoted as I. We assume that sensor-specific parameters (e.g., intrinsics, extrinsics, and distortion parameters) are known. For brevity, in the following text, the point cloud obtained from the camera is expressed directly in the IMU coordinate system, and coordinate transformation is omitted. The robot state $\mathbf{x}$ can be expressed as follows:

$$\mathbf{x} = \left[{}^G\boldsymbol{\phi}_I^\top, {}^G\mathbf{p}_I^\top, {}^G\mathbf{v}^\top, \mathbf{b}^{\mathbf{g}\top}, \mathbf{b}^{\mathbf{a}\top}\right]^\top \in \mathfrak{so}(3) \times \mathbb{R}^{12}$$

where the notations are explained in Nomenclature.

The proposed system is illustrated in Fig. 2. It processes multimodal information from three modules in an RGB-D camera: RGB images, depth images, and IMU measurements. This multimodal information is assumed to be time-synchronized. The system's objective is to estimate robot states using observations from these modules. The factor graph converts state estimation into a maximum a posteriori (MAP) problem. In Sections III-A–III-C, we present IMU state propagation and preintegration, introduce multimodal feature extraction and alignment, and discuss factor graph optimization and mapping.

A. IMU State Propagation and Preintegration

At the system's initialization, the robot remains stationary, allowing several accelerometer measurements to estimate gravity ${}^G\mathbf{g}$. Initially, both the gyroscope bias $\mathbf{b}^{\mathbf{a}}$ and the accelerometer bias $\mathbf{b}^{\mathbf{g}}$ are assumed to be zero, similar to FAST-LIO, and will be estimated later. The IMU measurements of angular velocity and acceleration are defined as follows:

$$\widehat{\boldsymbol{\omega}}_t = \boldsymbol{\omega}_t + \mathbf{b}^{\mathbf{g}}{}_t + \mathbf{n}^{\mathbf{g}}{}_t \tag{1}$$

$$\widehat{\mathbf{a}}_t = \mathrm{Exp}\left({}^I\boldsymbol{\phi}_{Gt}\right)\left(\mathbf{a}_t - {}^G\mathbf{g}\right) + \mathbf{b}^{\mathbf{a}}_t + \mathbf{n}^{\mathbf{a}}_t \tag{2}$$

where $\widehat{\boldsymbol{\omega}}_t$ and $\widehat{\mathbf{a}}_t$ represent the raw angular velocity and acceleration, respectively, in frame I at time t, affected by bias and white noise. ${}^I\boldsymbol{\phi}_{Gt}$ is the rotation from G to I.

Due to space limitations, only the state propagation and preintegration of continuous time are derived below; the discrete-time derivation is omitted. These measurements, $\widehat{\boldsymbol{\omega}}_t$ and $\widehat{\mathbf{a}}_t$, can be integrated to infer robot motion

$$\boldsymbol{\phi}_{t+\Delta t} = \mathrm{Log}(\mathrm{Exp}(\boldsymbol{\phi}_t)\,\mathrm{Exp}\left(\left(\hat{\boldsymbol{\omega}}_t - \mathbf{b}^{\omega}_t - \mathbf{n}^{\omega}_t\right)\Delta t\right)) \tag{3}$$

$$\mathbf{v}_{t+\Delta t} = \mathbf{v}_t + \mathbf{g}\Delta t + \mathrm{Exp}(\boldsymbol{\phi}_t)\left(\hat{\mathbf{a}}_t - \mathbf{b}^{\mathbf{a}}_t - \mathbf{n}^{\mathbf{a}}_t\right)\Delta t \tag{4}$$

$$\mathbf{p}_{t+\Delta t} = \mathbf{p}_t + \mathbf{v}_t\Delta t + \frac{1}{2}\mathbf{g}\Delta t^2 + \frac{1}{2}\mathrm{Exp}(\boldsymbol{\phi}_t)\left(\hat{\mathbf{a}}_t - \mathbf{b}^{\mathbf{a}}_t - \mathbf{n}^{\mathbf{a}}_t\right)\Delta t^2 \tag{5}$$

where $\boldsymbol{\phi} = {}^G\boldsymbol{\phi}_I$, $\mathbf{v} = {}^G\mathbf{v}_I$, $\mathbf{p} = {}^G\mathbf{p}_I$, $\mathbf{g} = {}^G\mathbf{g}$.

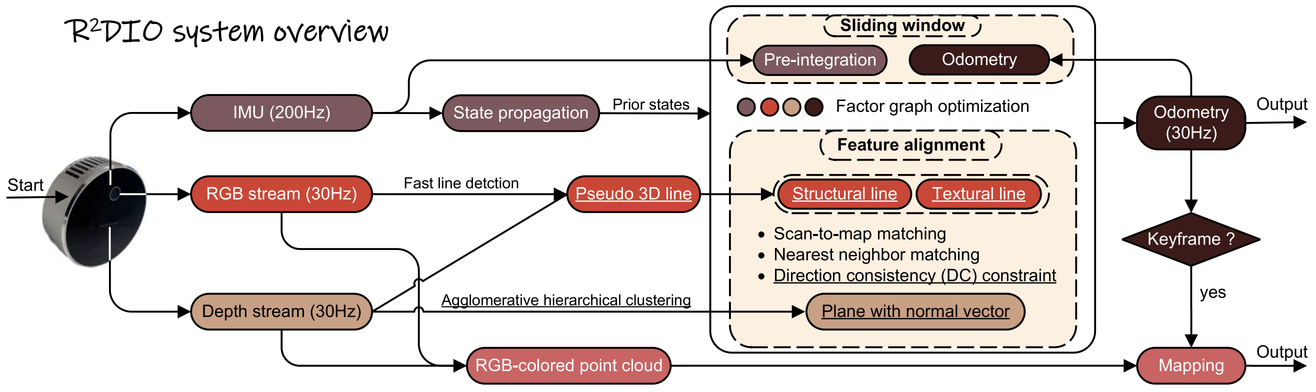


Fig. 2. R^{2}DIO comprises three main components: feature extraction and alignment, IMU preintegration, and state estimation. Its factor graph optimization incorporates four elements: IMU preintegration, historical odometry, line alignment, and plane alignment. Notably, the line and plane features utilized in this context exhibit superior quality compared to the edge and surface features typically found in traditional LiDAR SLAM. R^{2}DIO is an improvement upon SSL-SLAM3 in this work, the main innovative aspects are underscored for emphasis.

The IMU preintegration method, as introduced by Forster et al. [24], is employed to transform measurements taken between two time stamps i and j $(i < j)$ into a single generalized observation, ultimately generating a constraint between the corresponding states

$$\Delta\boldsymbol{\phi}_{ij} = \mathrm{Log}\big(\mathrm{Exp}(-\boldsymbol{\phi}_i)\,\mathrm{Exp}(\boldsymbol{\phi}_j)\big) \tag{6}$$

$$\Delta\mathbf{v}_{ij} = \mathrm{Exp}(-\boldsymbol{\phi}_i)\big(\mathbf{v}_j - \mathbf{v}_i - \mathbf{g}\Delta t_{ij}\big) \tag{7}$$

$$\Delta\mathbf{p}_{ij} = \mathrm{Exp}(-\boldsymbol{\phi}_i)\left(\mathbf{p}_j - \mathbf{p}_i - \mathbf{v}_i\Delta t_{ij} - \frac{1}{2}\mathbf{g}\Delta t_{ij}^2\right) \tag{8}$$

where $\mathrm{Exp}(-\boldsymbol{\phi}_i) = \mathrm{Exp}(\boldsymbol{\phi}_i)^{-1}$.

B. *Feature Extraction and Alignment*

Due to the notably lower vertical resolution of mechanical LiDAR compared to its horizontal resolution, LOAM and several similar systems extract edge and surf features by calculating the curvature of points within a ring. The uniform horizontal and vertical resolution of RGB and depth images from an RGB-D camera, however, allows the use of traditional image processing techniques for feature extraction. It is important to note that the proposed approach focuses on high-quality features corresponding to lines and planes, rather than edges and surfaces in LiDAR SLAM, which implies that utilizing the direction vectors of planes and lines can assist in feature matching to improve both accuracy and computational speed. Consequently, precise alignment can be achieved using a limited number of feature points. Furthermore, point cloud data lacks the proximity information that is present between pixels in color and depth maps, which represent spatial adjacency. This spatial adjacency can be effectively utilized for rapid clustering and feature extraction.

To accelerate the extraction process, an AHC method is employed. AHC algorithm effectively partitions pixels into distinct clusters without the need for a predefined number of clusters. It accomplishes feature clustering through a seed growing approach, leveraging the proximity of pixel features. Notably, the time complexity of the AHC algorithm is $O(n)$, indicating its high computational efficiency.

For line features, the AHC extractor proposed by Lee et al. [25] is employed. This approach initially detects

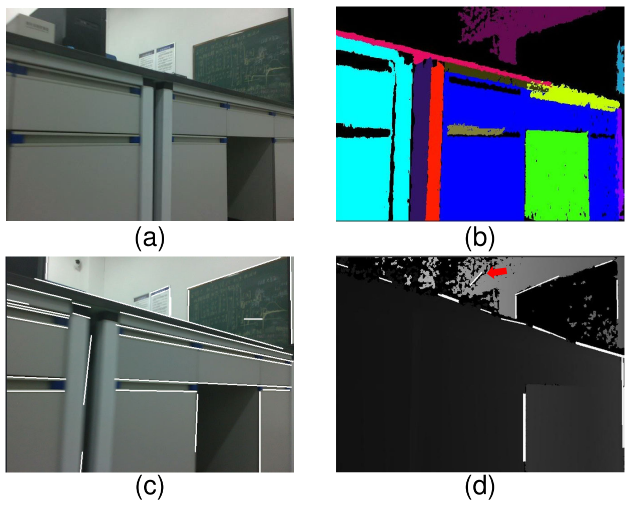


Fig. 3. Example of feature extraction is demonstrated in the subsequent figures. (a) RGB image depicting an office. (b) Plane detection in the depth image, where distinct colors signify separate planes. (c) Line detection in the RGB image. (d) Line detection in the depth image, with a red arrow indicating a line produced by noise.

Canny edges and subsequently connects two neighboring edge pixels to form a preliminary line segment. If this segment satisfies the collinearity criterion with other adjacent edge points, it expands by fitting the line segment to the subsequent edge pixel. The extractor iterates this procedure until all edge pixels have been assessed, ultimately returning line segments that surpass the established length threshold.

As depicted in Fig. 3, the AHC method can be employed to extract lines from both RGB and depth images. Nevertheless, only structural lines are present in the depth image, and some lines might be generated by noise, as demonstrated in Fig. 3(d). Conversely, lines extracted from the RGB image represent significant changes in both texture and structure. Due to the camera's limited FoV, textural features may be the only ones present within the FoV, with no structural features available. Relying exclusively on structural lines would result in system failure. To mitigate this issue, lines from RGB images are utilized, which we refer to as "pseudo 3-D lines." It is worth noting that in darker environments, lines extracted from the depth image may be more advantageous.

Given that the extrinsics of the depth and RGB modules are known, determining the depth value corresponding to pixel points on the line feature and their 3-D coordinates becomes a straightforward task. In order to address potential outliers for line features clustered by AHC, the random sample consensus (RANSAC) algorithm [26] is employed. This algorithm effectively filters out the outliers and enables the calculation of the parameters for pseudo 3-D lines, denoted as $\boldsymbol{L}$, based on the Plücker representation

$$\boldsymbol{L} = (\boldsymbol{l}^\top, \boldsymbol{m}^\top)^\top \tag{9}$$

where $\boldsymbol{l}$ is the direction unit vector of a line. $\boldsymbol{m} = \boldsymbol{l} \times \boldsymbol{p}$, where $\boldsymbol{p}$ is a point in the line.

For plane features, the AHC extractor [27] is applied to the depth image displayed in Fig. 3(c). Initially, the depth image is partitioned evenly into several smaller regions. Nodes belonging to the same plane are merged based on the coplanarity of adjacent points. Analogous to the line extractor, this procedure is iterated until all plane pixels have been assessed, and plane segments surpassing the number threshold are returned. RANSAC is also employed to fit these points, consequently obtaining accurate planes $\boldsymbol{P}$, represented by Hesse coordinates

$$\boldsymbol{P} = \left(\mathbf{n}_\pi^\top, d\right)^\top \tag{10}$$

where $\mathbf{n}_\pi$ denotes the normal unit vector of the plane, and d is the distance from the origin of the coordinate system to the plane.

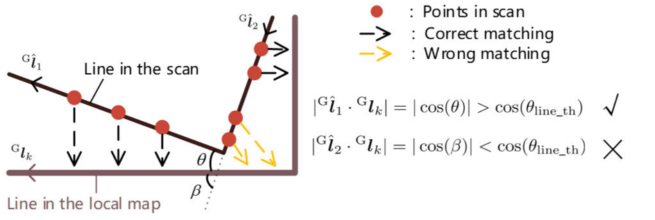
It should be noted that the threshold selection for AHC and RANSAC exhibits a certain level of robustness, often performing excellently across various indoor scenes with a fixed threshold. Furthermore, to maintain a similar level of real-time performance in pose estimation and prevent severe degradation due to an excessive number of features, we applied voxel filtering after feature extraction to adapt to varying feature quantities.

In our system, scan-to-map matching is utilized instead of scan-to-scan matching for aligning line and plane features. This approach can leverage more historical features and be constrained by features from earlier frames, substantially reducing long-term accumulation error. Point clouds at distant locations do not contribute to the current state estimation and increase computational expense. To mitigate this cost, the "map" refers to a local feature map that moves with the current frame and encompasses features in close proximity.

$\boldsymbol{p}_i^{\boldsymbol{L}}$ represents the points of the ith line feature, and $\boldsymbol{p}_j^{\boldsymbol{P}}$ denotes the points of the jth plane feature. These points ${}^G\boldsymbol{p}_i^{\boldsymbol{L}}$, ${}^G\boldsymbol{p}_j^{\boldsymbol{P}}$ in the global frame are employed to construct the line map and plane map, respectively, using the k-d tree. It is important to note that the directions of line and plane features, $\boldsymbol{l}$ and $\mathbf{n}_\pi$, are also transformed in the global map and recorded.

In this section, line feature alignment serves as an example to explain this process. The same methodology is applicable to plane feature alignment.

First, IMU measurements are employed to estimate the prior attitude and position $({}^G\hat{\boldsymbol{\phi}}_I, {}^G\hat{\mathbf{p}}_I)$ according to (6)–(8). Consequently, it is possible to transform the feature points ${}^I\boldsymbol{p}_i^{\boldsymbol{L}}$ from the current IMU frame to the global frame



: Points in scan
: Correct matching
: Wrong matching
${}^G\hat{\boldsymbol{l}}_2$
${}^G\hat{\boldsymbol{l}}_1$
Line in the scan
${}^G\boldsymbol{l}_k$
θ
β
Line in the local map
$|{}^G\hat{\boldsymbol{l}}_1 \cdot {}^G\boldsymbol{l}_k| = |\cos(\theta)| > \cos(\theta_{\text{line\_th}})$ ✓
$|{}^G\hat{\boldsymbol{l}}_2 \cdot {}^G\boldsymbol{l}_k| = |\cos(\beta)| < \cos(\theta_{\text{line\_th}})$ ✗

Fig. 4. Relying solely on nearest neighbor matching may lead to incorrect matching, as shown by the yellow arrows.

$${}^G\hat{\boldsymbol{p}}_i^{\boldsymbol{L}} = \text{Exp}({}^G\hat{\boldsymbol{\phi}}_I)^I\boldsymbol{p}_i^{\boldsymbol{L}} + {}^G\hat{\mathbf{p}}_I. \tag{11}$$

Furthermore, the direction ${}^I\boldsymbol{l}$ can be transformed as well

$${}^G\hat{\boldsymbol{l}} = \text{Exp}\left({}^G\hat{\boldsymbol{\phi}}_I\right)^I\boldsymbol{l} \tag{12}$$

Then, five nearest points ${}^G\boldsymbol{p}_k^{\boldsymbol{L}}$ in the k-d tree of the line feature map can be selected, where $k = 1, \ldots, 5$. ${}^G\boldsymbol{l}_k$ represents the direction vector of the nearest point.

The method of nearest-neighbor matching often results in incorrect matches in Fig. 4. For example, a point on a horizontal line may be matched with a point on a vertical line, which is clearly a mismatch. To address this issue, we apply the feature DC constraint. If the direction ${}^G\hat{\boldsymbol{l}}$ of the lines corresponding to the current point ${}^G\hat{\boldsymbol{p}}_i^{\boldsymbol{L}}$ and the direction ${}^G\boldsymbol{l}_k$ corresponding to the nearest points ${}^G\boldsymbol{p}_k^{\boldsymbol{L}}$ are significantly different, which indicates that

$$|{}^G\hat{\boldsymbol{l}} \cdot {}^G\boldsymbol{l}_k| = |\cos(\theta)| < \cos(\theta_{\text{line\_th}}). \tag{13}$$

${}^G\hat{\boldsymbol{p}}_i^{\boldsymbol{L}}$ and ${}^G\boldsymbol{p}_k^{\boldsymbol{L}}$ will be considered mismatches. In (13), the angle threshold, $\theta_{\text{line\_th}}$, is manually set and initially configured to be around 20°. We determine whether two lines are close by evaluating the magnitude of the dot product of their unit direction vectors. As the optimization iterations proceed, the pose estimation becomes progressively more accurate, resulting in a gradual reduction in the threshold $\theta_{\text{line\_th}}$ with each iteration, which implies a stricter requirement for directional consistency. The selection of the threshold is related to the accuracy of the IMU. Due to the feature DC constraint, many mismatched points are eliminated before state estimation, significantly improving the real-time capability and accuracy.

Remark: Setting the threshold $\theta_{\text{line\_th}}$ too large, such as 60°, diminishes the effectiveness of the constraint. Conversely, setting it too small, like 2°, results in an insufficiency of point-line and point-plane correspondences, which prevents ICP convergence. After conducting a series of experiments, we determined the initial value to be 20°, progressively reducing it to 10° and subsequently 5° over the following two iterations. What's more, if the IMU has high accuracy, its predicted pose will be more precise, allowing us to use a smaller threshold in the first iteration to accelerate convergence. Overall, the currently set threshold is capable of accommodating more typical indoor scenarios.

In addition, since the extraction of line features and plane features is independent, it can be executed in two separate threads to increase the speed of feature extraction.

C. Factor Graph Optimization and Mapping

We estimate the current state and optimize historical states within the sliding window based on the factor graph. There are four factors depicted in Fig. 2: line feature, plane feature, preintegration, and odometry.

1) Residual From the Line Factor: The cost function is constructed based on the point-to-line metric. Initially, points $\boldsymbol{p^L}_i$ are sampled on lines of the current frame i. For each point, it is transformed into the global frame using (11) and its five nearest line points ${}^G\boldsymbol{p}_k^{\boldsymbol{L}}$ in the local feature map are searched, where $k = 1, \ldots, 5$. If they satisfy the DC constraint, the mean value ${}^G\overline{\boldsymbol{p}}^{\boldsymbol{L}}$ and covariance matrix of the five points' coordinates are computed. Eigen decomposition is used to obtain the eigenvalues, assuming the largest eigenvalue is significantly larger than the others. In this case, these five points approximately form a line segment, with its direction vector $\mathbf{n}^{\boldsymbol{L}}$ being the eigenvector corresponding to the largest eigenvalue. Thus, the line segment can be represented by two points: ${}^G\boldsymbol{p}_1^{\boldsymbol{L}} = {}^G\overline{\boldsymbol{p}}^{\boldsymbol{L}} + \delta\mathbf{n}^{\boldsymbol{L}}$ and ${}^G\boldsymbol{p}_2^{\boldsymbol{L}} = {}^G\overline{\boldsymbol{p}}^{\boldsymbol{L}} - \delta\mathbf{n}^{\boldsymbol{L}}$. The residual $\mathbf{r}_{\boldsymbol{L}}$ from the line factor is given by

$$\mathbf{r}_{\boldsymbol{L}} = \frac{\left\| \left({}^G\overline{\boldsymbol{p}}^{\boldsymbol{L}} - {}^G\boldsymbol{p}_1^{\boldsymbol{L}}\right) \times \left({}^G\overline{\boldsymbol{p}}^{\boldsymbol{L}} - {}^G\boldsymbol{p}_2^{\boldsymbol{L}}\right) \right\|}{\left\| {}^G\boldsymbol{p}_1^{\boldsymbol{L}} - {}^G\boldsymbol{p}_2^{\boldsymbol{L}} \right\|}. \tag{14}$$

2) Residual From the Plane Factor: Similar to the line factor, a cost function based on the point-to-plane metric is constructed. After finding the nearest five plane points that satisfy the DC constraint, they can be fitted to a plane using QR decomposition, and the normal vector is estimated as $\mathbf{n}^{\boldsymbol{P}}$. The residual can be expressed as

$$\mathbf{r}_{\boldsymbol{P}} = \frac{\left\| \mathbf{n}^{\boldsymbol{P}} \cdot {}^G\boldsymbol{p}^{\boldsymbol{P}} + 1 \right\|}{\left\| \mathbf{n}^{\boldsymbol{P}} \right\|}. \tag{15}$$

In particular, we have strived to construct the cost function using line-to-line and plane-to-plane residuals, as opposed to point-to-line and point-to-plane residuals. Unfortunately, given the limited accuracy of line feature extraction and the prevalence of mismatches, a more robust methodology is required, even though line-to-line and plane-to-plane residuals prove highly efficient for real-time optimization.

3) Residual From the IMU Preintegration Factor: The IMU preintegration converts a series of instantaneous measurements between two frames i and j into a generalized observation, constraining the two states with respect to their attitude, position, and velocity

$$\mathbf{r}_{I_i}^{\ j} = \begin{bmatrix} \operatorname{Log}\left(\operatorname{Exp}(-\boldsymbol{\phi}_i)\operatorname{Exp}(\boldsymbol{\phi}_j)\right) - \Delta\boldsymbol{\phi}_{ij} \\ \operatorname{Exp}(-\boldsymbol{\phi}_i)\left(\mathbf{v}_j + \mathbf{g}\Delta t - \mathbf{v}_i\right) - \Delta\mathbf{v}_{ij} \\ \operatorname{Exp}(-\boldsymbol{\phi}_i)\left(\mathbf{p}_j - \mathbf{p}_i + \frac{1}{2}\mathbf{g}\Delta t^2 - \mathbf{v}_i\Delta t\right) - \Delta\mathbf{p}_{ij} \\ \mathbf{b}_j^{\mathbf{g}} - \mathbf{b}_i^{\mathbf{g}} \\ \mathbf{b}_j^{\mathbf{a}} - \mathbf{b}_i^{\mathbf{a}} \end{bmatrix} \tag{16}$$

where $\mathbf{r}Ii^j$ is the residual from the IMU, $\boldsymbol{\phi} = {}^G\boldsymbol{\phi}_I$, $\mathbf{v} = {}^G\mathbf{v}_I$, $\mathbf{p} = {}^G\mathbf{p}_I$, and $\mathbf{g} = {}^G\mathbf{g}$. Consequently, the IMU measurements and other constraints can be utilized to jointly estimate the state, resulting in a tightly coupled system.

4) Residual From the Odometry Factor: The sliding window provides valuable information for state estimation and track smoothing, while information outside the window is largely irrelevant. Moreover, calculating residuals using all line and plane features in the window may be inefficient. To balance real-time performance and accuracy, the pose transformation between two historical keyframes, denoted as $(\hat{\boldsymbol{\phi}}_{i-1}^i, \hat{\mathbf{p}}_{i-1}^i)$, is used as the odometry constraint. Specifically, $\hat{\boldsymbol{\phi}}_{i-1}^i = \operatorname{Log}(\operatorname{Exp}(\hat{\boldsymbol{\phi}}_{\mathrm{i-1}})\operatorname{Exp}(-\hat{\boldsymbol{\phi}}_I))$ and $\hat{\mathbf{p}}_{i-1}^i = \operatorname{Exp}(\hat{\boldsymbol{\phi}}_{\mathrm{i-1}})(\hat{\mathbf{p}}_i - \hat{\mathbf{p}}_{i-1})$. It is essential to note that $(\hat{\boldsymbol{\phi}}_{i-1}^i, \hat{\mathbf{p}}_{i-1}^i)$ is the result obtained via factor graph optimization for the first time and remains constant throughout the optimization process. The residual can be computed as

$$\mathbf{r}_{o_{i-1}}^{\ i} = \begin{bmatrix} \operatorname{Log}\left(\operatorname{Exp}\left(-\hat{\boldsymbol{\phi}}_{i-1}^i\right)\operatorname{Exp}\left(-\boldsymbol{\phi}_{\mathrm{i-1}}\right)\operatorname{Exp}\left(\boldsymbol{\phi}_i\right)\right) \\ \operatorname{Exp}\left(-\hat{\boldsymbol{\phi}}_{i-1}^i\right)\left(\operatorname{Exp}\left(\boldsymbol{\phi}_{\mathrm{i-1}}\right)(\mathbf{p}_i - \mathbf{p}_{i-1}) - \hat{\mathbf{p}}_{i-1}^i\right) \end{bmatrix} \tag{17}$$

where $\boldsymbol{\phi}_i$ and $\mathbf{p}_i$ are the variables to be optimized.

5) Factor Graph Optimization: After calculating the residuals, factor graph opti0mization is employed to estimate the states $\{\mathbf{x}_{i-n}, \ldots, \mathbf{x}_i\}$ within the $n+1$ frame sliding window. This is accomplished by minimizing the sum of the line and plane residuals, the IMU preintegration residuals, and the odometry residuals. The optimization results in a maximum posterior estimation given by

$$\min_{\{\mathbf{x}_{i-n},\ldots,\mathbf{x}_i\}} \Bigg\{ \sum_{\boldsymbol{p}^L \in \{\boldsymbol{p}^L\}_i} \rho\left(\|\mathbf{r}_{\boldsymbol{L}}\|_{\sigma_L}^2\right) + \sum_{\boldsymbol{p}^P \in \{\boldsymbol{p}^P\}_i} \rho\left(\|\mathbf{r}_{\boldsymbol{P}}\|_{\sigma_P}^2\right) + \sum_{k=i-n}^{i} \left\|\mathbf{r}_{I_{k-1}}^{\ k}\right\|_{\sigma_I}^2 + \sum_{k=i-n}^{i-1} \left\|\mathbf{r}_{o_{k-1}}^{\ k}\right\|_{\sigma_o}^2 \Bigg\}. \tag{18}$$

In the optimization equation, σ represents the covariance that determines the weights in the cost function, while $\rho(\cdot)$ denotes the robust kernel function used to mitigate the influence of outliers. In this article, the Huber kernel is used. Due to space constraints, the derivation of the Jacobian formula is omitted. The factor graph optimization can be solved using the Levenberg–Marquardt algorithm, which is implemented in Ceres Solver [28].

6) Mapping: The colored point cloud is generated by combining RGB images and depth images using the correct camera intrinsics. To eliminate noise, any point whose distance from its adjacent points exceeds a certain threshold is removed. Additionally, to optimize efficiency, a new keyframe is created and the point cloud is registered to the map only when the motion exceeds a specific angle or distance threshold. Finally, the generated point cloud is registered with the world coordinate system based on the previously estimated pose to complete incremental mapping.

IV. Experiments Evaluation

A. Experiment Setup

The experimental setup comprises two sets of devices as illustrated in Fig. 5 for Accuracy Evaluation I and Fig. 6 for Accuracy Evaluation II. The hand-held experimental device depicted in Fig. 6 includes three cameras, a LiDAR, and a mini PC. Our system operates on a low-power Intel Core i5-1135G7@2.4-GHz CPU. It is noteworthy that the

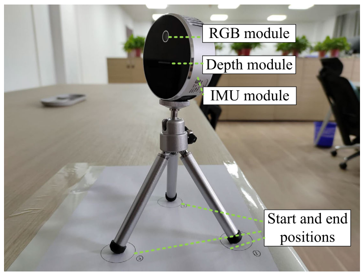


Fig. 5. Intel RealSense L515, a ToF RGB-D camera with a built-in IMU module, was used for Accuracy Evaluation I. The three circles indicate the locations of the start and end points.

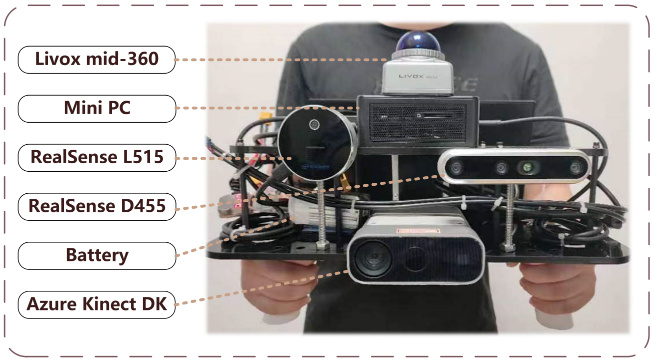


Fig. 6. Our hand-held experimental device for Accuracy Evaluation II. Only the Azure Kinect DK was used to applied during the experiment. Additionally, Livox mid-360 was employed alongside FAST-LIO as a device for acquiring ground truth data.

RealSense L515 and Azure Kinect DK represent ToF depth cameras, while the RealSense D455 is a structured light depth camera. ToF cameras offer significantly higher depth accuracy compared to structured light cameras. Given that our algorithm heavily relies on depth accuracy, it delivers optimal performance when run on a ToF RGB-D camera. Due to the subpar performance of RealSense D455 in our experiments, it will not be included in further discussion.

Due to the unavailability of suitable open-source datasets, we collected our own data and performed a simple accuracy evaluation. Additionally, we conducted experiments using a hand-held device to further demonstrate the robustness of our system. These experiments presented greater challenges compared to those conducted on mobile robots, as the hand-held device is susceptible to vibration and large changes in viewing angle. Mobile robots typically exhibit limited motion with only three degrees of freedom in planar movement. We divided the experiments into two groups and evaluated the accuracy and robustness from the perspectives of end-to-end and comparison with ground truth.

B. Accuracy Evaluation I: End-to-End Error

First, we evaluated the accuracy with RealSense L515 in Fig. 5 by calculating the end-to-end error, which is the distance between the estimated trajectory's start and end points. The ground truth of the error is zero. We conducted three experiments in three indoor scenes: an office, a hall, and a display board (shown in Fig. 7). We compared the accuracy and robustness of R^{2}DIO, R^{2}DIO (w/o DC), R^{2}DIO (singe modal), SSL-SLAM, ORB-SLAM2, and others, by running them on different routes with a hand-held platform in each scene. Here, R^{2}DIO (singe modal) refers to extracting line features from depth maps instead of color images. Since the RGB-D-inertial mode of ORB-SLAM3 requires complex initialization and cannot estimate position and pose from the beginning, we could not evaluate its accuracy using end-to-end error. Instead, we used the RGB-D mode of ORB-SLAM3. For the same reason, we tested Vins-RGBD instead of Vins-fusion.

TABLE I
SYSTEM ACCURACY COMPARISON (END-TO-END ERROR IN m)

Approach	Office			Exhibition hall			Display board		
	I	II	III	I	II	III	I	II	III
Total length	20.5	18.9	30.2	40.1	28.0	20.5	6.1	5.9	10.3
Ours	0.24	0.05	**0.31**	**0.19**	0.18	0.32	**0.03**	0.03	**0.04**
Ours(w/o DC)	0.50	0.31	0.55	1.89	0.84	2.34	2.11	0.42	1.64
Ours(single modal)	2.77	1.23	0.87	2.45	0.63	1.32	3.25	0.05	1.89
SSL-SLAM3	8.01	4.16	3.33	2.30	3.62	3.57	2.55	0.03	2.40
ORB-SLAM2	Fail	Fail	Fail	1.08	**0.02**	0.03	**0.03**	0.12	0.05
ORB-SLAM3	Fail	Fail	Fail	1.15	**0.02**	**0.02**	0.04	**0.02**	**0.04**
VINS-RGBD	1.02	0.39	3.33	3.20	0.03	0.05	0.10	0.06	0.12
VoxelMap	**0.02**	**0.01**	0.35	**0.19**	2.40	0.04	1.19	5.06	3.22

All three scenarios present challenges for SLAM: in the office, a large white wall may appear in the camera's view; in the hall, there may be a lack of structural features in the FoV; facing the display board, there may be a shortage of structural features but an abundance of textural features. The trajectories of the experiments are shown in Fig. 8 and Table I. Due to the large number of tracks, we only plotted the trajectories of three systems, R^{2}DIO, SSL-SLAM, and ORB-SLAM3 in Fig. 8 for clarity. Note that if ORB-SLAM3 fails to track the local map, it will create a new map. We only plotted the trajectory before tracking failed.

As indicated in Table I, the trajectories estimated by R^{2}DIO are accurate, with the distance between the start and end positions being close to the truth-value of zero. In Fig. 8(a) and (b) of office, ORB-SLAM3 failed due to the large white wall in view, and there was apparent drift in the trajectories of SSL-SLAM because it only used structural features. Compared with R^{2}DIO with single modal in Table I, R^{2}DIO showed good accuracy and robustness due to its use of multimodal constraints. In Fig. 8(g) and (i) of display board, SSL-SLAM failed to estimate quickly because it relied on structural constraints in a single direction. Conversely, R^{2}DIO was able to effectively utilize the textural line features on the display board. It is worth mentioning that VoxelMap exhibits high accuracy when the system does not experience drift. This should be attributed to its modeling of plane errors, which can be used for learning and reference in the future.

C. Accuracy Evaluation II: Compared With Ground Truth

LiDAR-inertial odometry using a large FoV LiDAR, such as the Livox Mid-360 shown in Fig. 6, has demonstrated high accuracy and consistent mapping performance indoors.

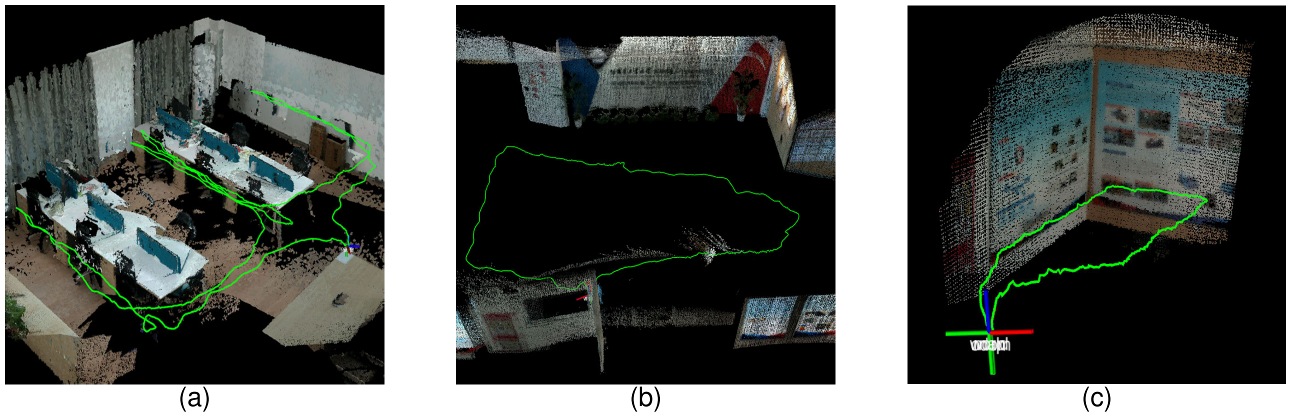


Fig. 7. Dense maps were constructed by R^{2}DIO with RealSense L515 in (a) office, (b) hall, and (c) display board using a hand-held device. The trajectory is depicted in green. A link to the video can be found at the bottom of page 1.

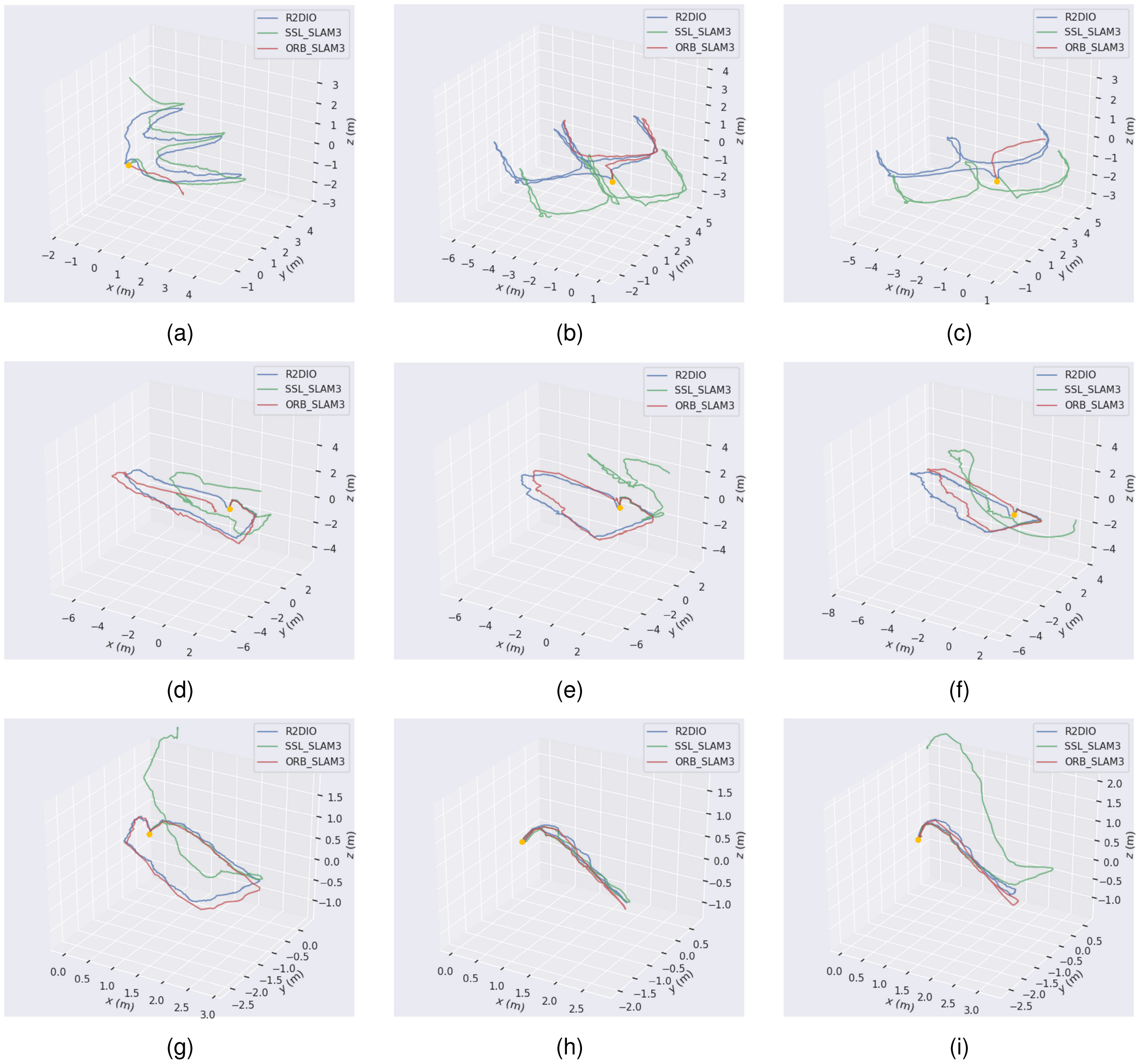


Fig. 8. Trajectories of three systems were estimated in three different scenes. (a)–(c) Office. (d)–(f) Hall. (g)–(i) Facing a display board. The starting point is indicated by a yellow dot. The results were obtained using an open-source evaluation tool, evo [29].

TABLE II

Performance Comparison in Different Indoor Environments (RMSE ATE in cm). Results Reported By the Papers of Each System, for All the Frames in the Trajectory, Comparing With the Processed GT

Seq.	Total length	Ours	Ours (w/o DC)	Ours (singe modal)	SSL-SLAM3	ORB-SLAM2	ORB-SLAM3	VINS-RGBD	VoxelMap
Office I	30.545	0.115	0.438	0.878	1.013	0.718	<u>0.036</u>	0.059	**0.024**
Office II	27.537	**0.023**	0.138	0.315	0.228	Fail	Fail	3.098	<u>0.025</u>
Office III	42.341	<u>0.042</u>	0.250	0.449	0.465	0.119	0.093	0.339	**0.012**
Office IV	29.324	**0.021**	<u>0.198</u>	0.684	0.673	Fail	Fail	3.236	0.218
Exhibition hall I	33.980	**0.035**	<u>0.089</u>	1.092	1.876	0.125	0.143	0.312	1.018
Exhibition hall II	29.087	0.143	1.980	0.789	1.769	<u>0.035</u>	**0.022**	0.235	1.718
Exhibition hall III	20.877	<u>0.037</u>	0.402	0.172	0.340	**0.024**	<u>0.037</u>	0.323	0.118

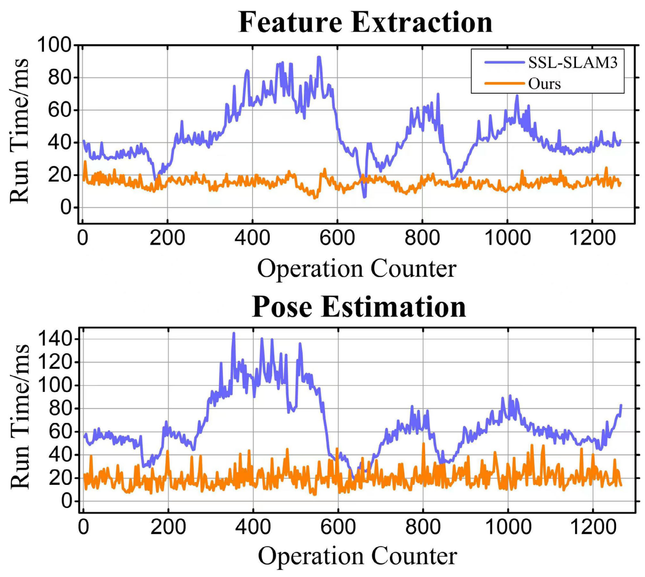


Fig. 9. Time cost of each operation is compared with SSL-SLAM. The feature extraction module in our approach achieves twice the speed, and the state estimation module accelerates threefold.

In this evaluation experiment, we used the trajectory estimated by FAST-LIO [11] as the ground truth. We recorded seven sequences in an office and exhibition hall environment with Azure Kinect DK. Similar to Accuracy Evaluation I's scene, the office has rich structural features but lacks texture features due to the predominance of white walls in the FoV. The exhibition hall scene is relatively spacious with rich texture and patterns but lacks structural features. As shown in Table II, ORB-SLAM2, ORB-SLAM3, and VINS-RGBD achieved excellent location accuracy in the office based on texture features, while SSL-SLAM3 and VoxelMap achieved excellent location accuracy in the exhibition hall based on structural features. However, in other scenarios, they experienced failures or significant drift.

To sum up, the proposed system combines the advantages of both texture and structural features, exhibiting robust and high-precision performance in both scenarios. The ablation experiments presented in Tables I and II demonstrate the effectiveness of DC constraints in improving accuracy and the benefits of multimodal constraints in enhancing robustness.

D. Real-Time Performance Evaluation

The feature extraction and alignment processes in our system were derived from SSL-SLAM3. Due to the superior feature extraction, our system can estimate high-precision poses using fewer features. Furthermore, the integration of feature direction constraint facilitates the removal of numerous mismatched features, substantially decreasing the time needed for state estimation. Fig. 9 illustrates the modular time costs associated with feature extraction and state estimation. Although many RGB-D cameras support a 30-Hz image recording rate, our system operates at this frequency, surpassing SSL-SLAM3's 10-Hz performance. As a result, our system exhibits increased robustness in fast-motion situations.

Remark: In the Real-Time Performance Evaluation, we used color and depth images with a resolution of 640 × 480 and a capture frequency of 30 Hz. For each image, R^{2}DIO processes and estimates the pose, whereas SSL-SLAM3, with its limited real-time capabilities, processes every third frame. Throughout the experiment, we maintained consistency in other aspects, such as the keyframe strategy and dense mapping resolution. Furthermore, we conducted tests in multiple scenarios, which confirmed the stable operation of R^{2}DIO at a speed of 30 Hz.

V. Discussion

The current version of R^{2}DIO lacks a loop closure module, resulting in an accumulation of odometry errors. In contrast, ORB-SLAM3 [2] utilizes ORB features to align short-term and mid-term data and executes loop-closing for long-term data association, thereby achieving superior positioning accuracy in experimental settings like Hall II and III. It is thus hypothesized that the integration of ORB features into R^{2}DIO for mid-term and long-term data association could enhance its accuracy.

Moreover, the experimental findings indicate that the probability map approach of VoxelMap [13] boasts higher precision and can exhaustively exploit structural information, warranting further investigation and refinement. The experiment also revealed that the map objects created by R^{2}DIO often exhibit abrupt color changes, and color information is not used to constrain states. Inspired by R^{3}LIVE++ [30], we propose the use of radiance information to construct a radiance map. By integrating color information into the state estimation process, this method could improve both the visual quality and the accuracy of the generated maps.

VI. Conclusion

To ensure a robot's capability to perform SLAM in challenging scenarios, our system utilizes advanced feature extraction methods for both RGB and depth images. Besides nearest neighbor matching, we incorporate DC constraint to effectively eliminate mismatches. Our compelling experimental results show that these enhancements substantially improve real-time performance, enabling the system to operate at 30 Hz on a microcomputer. Furthermore, by integrating multimodal constraints, such as line features in RGB images, plane features in depth images, IMU preintegration factors, and odometry factors, our system exhibits remarkable robustness and accuracy. Consequently, R^{2}DIO is readily implementable on various indoor mobile platforms, representing a significant advancement in robotic navigation and manipulation.

Acknowledgment

The authors would like to express our gratitude to the authors of SSL-SLAM, ORB-SLAM3, FAST-LIO, VoxelMap, and other frameworks for their commitment to open-source work.

References

[1] R. Mur-Artal and J. D. Tardós, "ORB-SLAM2: An open-source SLAM system for monocular, stereo, and RGB-D cameras," *IEEE Trans. Robot.*, vol. 33, no. 5, pp. 1255–1262, Oct. 2017.

[2] C. Campos, R. Elvira, J. J. G. Rodriguez, J. M. M. Montiel, and J. D. Tardos, "ORB-SLAM3: An accurate open-source library for visual, visual–inertial, and multimap SLAM," *IEEE Trans. Robot.*, vol. 37, no. 6, pp. 1874–1890, Dec. 2021.

[3] A. Dai, M. Nießner, M. Zollhöfer, S. Izadi, and C. Theobalt, "BundleFusion: Real-time globally consistent 3D reconstruction using on-the-fly surface re-integration," *ACM Trans. Graph.*, vol. 36, no. 4, p. 1, Jul. 2017.

[4] H. Wang, C. Wang, and L. Xie, "Lightweight 3-D localization and mapping for solid-state LiDAR," *IEEE Robot. Autom. Lett.*, vol. 6, no. 2, pp. 1801–1807, Apr. 2021.

[5] J. Zhang and S. Singh, "LOAM: LiDAR odometry and mapping in real-time," in *Proc. Robot., Sci. Syst. X*, Jul. 2014, pp. 1–9.

[6] D. Rozenberszki and A. L. Majdik, "LOL: LiDAR-only odometry and localization in 3D point cloud maps," in *Proc. IEEE Int. Conf. Robot. Autom. (ICRA)*, May 2020, pp. 4379–4385.

[7] D. Chetverikov, D. Svirko, D. Stepanov, and P. Krsek, "The trimmed iterative closest point algorithm," in *Proc. Object Recognit. Supported User Interact. Service Robots*, Aug. 2002, pp. 545–548.

[8] H. Wang, C. Wang, C.-L. Chen, and L. Xie, "F-LOAM: Fast LiDAR odometry and mapping," in *Proc. IEEE/RSJ Int. Conf. Intell. Robots Syst. (IROS)*, Sep. 2021, pp. 4390–4396.

[9] T. Shan, B. Englot, D. Meyers, W. Wang, C. Ratti, and D. Rus, "LIO-SAM: Tightly-coupled LiDAR inertial odometry via smoothing and mapping," in *Proc. IEEE/RSJ Int. Conf. Intell. Robots Syst. (IROS)*, Oct. 2020, pp. 5135–5142.

[10] W. Xu and F. Zhang, "FAST-LIO: A fast, robust LiDAR-inertial odometry package by tightly-coupled iterated Kalman filter," *IEEE Robot. Autom. Lett.*, vol. 6, no. 2, pp. 3317–3324, Apr. 2021.

[11] W. Xu, Y. Cai, D. He, J. Lin, and F. Zhang, "FAST-LIO2: Fast direct LiDAR-inertial odometry," *IEEE Trans. Robot.*, vol. 38, no. 4, pp. 2053–2073, Aug. 2022.

[12] C. Qin, H. Ye, C. E. Pranata, J. Han, S. Zhang, and M. Liu, "LINS: A LiDAR-inertial state estimator for robust and efficient navigation," in *Proc. IEEE Int. Conf. Robot. Autom. (ICRA)*, May 2020, pp. 8899–8906.

[13] C. Yuan, W. Xu, X. Liu, X. Hong, and F. Zhang, "Efficient and probabilistic adaptive voxel mapping for accurate online LiDAR odometry," *IEEE Robot. Autom. Lett.*, vol. 7, no. 3, pp. 8518–8525, Jul. 2022.

[14] S. Izadi et al., "KinectFusion: Real-time 3D reconstruction and interaction using a moving depth camera," in *Proc. 24th Annu. ACM Symp. User Interface Softw. Technol.*, Oct. 2011, pp. 559–568.

[15] T. Whelan, R. F. Salas-Moreno, B. Glocker, A. J. Davison, and S. Leutenegger, "ElasticFusion: Real-time dense SLAM and light source estimation," *Int. J. Robot. Res.*, vol. 35, no. 14, pp. 1697–1716, Dec. 2016.

[16] J. Engel, V. Koltun, and D. Cremers, "Direct sparse odometry," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 40, no. 3, pp. 611–625, Mar. 2018.

[17] Z. Yuan, K. Cheng, J. Tang, and X. Yang, "RGB-D DSO: Direct sparse odometry with RGB-D cameras for indoor scenes," *IEEE Trans. Multimedia*, vol. 24, pp. 4092–4101, 2022.

[18] L. Wang and Z. Wu, "RGB-D SLAM with Manhattan frame estimation using orientation relevance," *Sensors*, vol. 19, no. 5, p. 1050, Mar. 2019.

[19] Y. Li, R. Yunus, N. Brasch, N. Navab, and F. Tombari, "RGB-D SLAM with structural regularities," in *Proc. IEEE Int. Conf. Robot. Autom. (ICRA)*, May 2021, pp. 11581–11587.

[20] Z. Shan, R. Li, and S. Schwertfeger, "RGBD-inertial trajectory estimation and mapping for ground robots," *Sensors*, vol. 19, no. 10, p. 2251, May 2019.

[21] J. Liu, X. Li, Y. Liu, and H. Chen, "RGB-D inertial odometry for a resource-restricted robot in dynamic environments," *IEEE Robot. Autom. Lett.*, vol. 7, no. 4, pp. 9573–9580, Oct. 2022.

[22] T. Qin, P. Li, and S. Shen, "VINS-mono: A robust and versatile monocular visual-inertial state estimator," *IEEE Trans. Robot.*, vol. 34, no. 4, pp. 1004–1020, Aug. 2018.

[23] H. Wang. (2022). *sslSLAM3*. [Online]. Available: https://github.com/wh200720041/ssl_slam3

[24] C. Forster, L. Carlone, F. Dellaert, and D. Scaramuzza, "On-manifold preintegration for real-time visual-inertial odometry," *IEEE Trans. Robot.*, vol. 33, no. 1, pp. 1–21, Feb. 2017.

[25] J. H. Lee, S. Lee, G. Zhang, J. Lim, W. K. Chung, and I. H. Suh, "Outdoor place recognition in urban environments using straight lines," in *Proc. IEEE Int. Conf. Robot. Autom. (ICRA)*, May 2014, pp. 5550–5557.

[26] M. A. Fischler and R. Bolles, "Random sample consensus: A paradigm for model fitting with applications to image analysis and automated cartography," *Commun. ACM*, vol. 24, no. 6, pp. 381–395, 1981.

[27] C. Feng, Y. Taguchi, and V. R. Kamat, "Fast plane extraction in organized point clouds using agglomerative hierarchical clustering," in *Proc. IEEE Int. Conf. Robot. Autom. (ICRA)*, May 2014, pp. 6218–6225.

[28] S. Agarwal, K. Mierle, and T. C. S. Team. (Mar. 2022). *Ceres Solver*. [Online]. Available: https://github.com/ceres-solver/ceres-solver

[29] M. Grupp. (2017). *EVO: Python Package for the Evaluation of Odometry and SLAM*. [Online]. Available: https://github.com/MichaelGrupp/evo

[30] J. Lin and F. Zhang, "R^{3}LIVE++: A robust, real-time, radiance reconstruction package with a tightly-coupled LiDAR-inertial-visual state estimator," Sep. 2022, *arXiv:2209.03666*.

Jie Xu received the B.E. and M.S. degrees from the Harbin Institute of Technology, Harbin, China, in 2018 and 2020, respectively, where he is currently pursuing the Ph.D. degree with the School of Mechanical and Electrical Engineering.

He is an intern at Yangtze River Delta HIT Robot Technology Research Institute. His research interests include multisensor calibration, RGB-D SLAM, and multimodal SLAM.

Ruifeng Li received the Ph.D. degree from the Harbin Institute of Technology, Harbin, China, in 1996.

He is currently a Professor and the Vice Director of the State Key Laboratory of Robotics and System, Harbin Institute of Technology. His research interests include artificial intelligence and robotics.

Dr. Li is a member of the Chinese Association for Artificial Intelligence.

Song Huang received the B.E. degree from Anhui Polytechnic University (AHPU), Wuhu, China, in 2021. He is currently pursuing the M.E. degree with Anhui Normal University, Wuhu.

He is an intern at Yangtze River Delta HIT Robot Technology Research Institute. His research interests include multisensor fusion for localization and mapping, and sensor calibration.

Xiongwei Zhao received the M.S. degree from the University of Science and Technology Beijing, Beijing, China, in 2021. He is currently pursuing the Ph.D. degree with the School of Electronic Information, Harbin Institute of Technology (Shenzhen), Shenzhen, China.

His research interests include state estimation with multisensor fusion, place recognition, and multisensor calibration.

Shuxin Qiu is currently pursuing the master's degree with the School of Mechanical Engineering, Nanchang Institute of Technology, Nanchang, China.

He is an intern at Yangtze River Delta HIT Robot Technology Research Institute. His research interests include dynamic simultaneous localization and mapping (Dynamic-SLAM) and embedded device development.

Zhijun Chen received the M.E. degree in photogrammetry and remote sensing from Hohai University (HHU), Nanjing, China, in 2016.

He is currently with the Yangtze River Delta HIT Robot Technology Research Institute, Wuhu, China. His main research interests include LiDAR simultaneous localization and mapping (LiDAR-SLAM).

Lijun Zhao (Member, IEEE) received the Ph.D. degree in mechatronic engineering from the Institute of Robotics, Harbin Institute of Technology, Harbin, China, in 2009.

He is currently a Professor with the State Key Laboratory of Robotics and System, Harbin Institute of Technology. His current research interests include robot localization, unknown environment mapping, and navigation. He is also the Executive Dean of the Yangtze River Delta HIT Robot Technology Research Institute.