electronics

MDPI

Article

Mergeable Probabilistic Voxel Mapping for LiDAR–Inertial–Visual Odometry

Balong Wang [1], Nassim Bessaad [1], Huiying Xu [1,*], Xinzhong Zhu [1,2,3] and Hongbo Li [3]

[1] School of Computer Science and Technology of Zhejiang Normal University, Jinhua 321004, China; wbl0525@zjnu.edu.cn (B.W.); bessaadnassim@yahoo.com (N.B.); zxz@zjnu.edu.cn (X.Z.)
[2] Zhejiang Key Laboratory of Intelligent Education Technology and Application, Zhejiang Normal University, Jinhua 321004, China
[3] Beijing Geekplus Technology Co., Ltd., Beijing 100101, China; jason.li@geekplus.com
* Correspondence: xhy@zjnu.edu.cn

Abstract: To address the limitations of existing LiDAR–visual fusion methods in adequately accounting for map uncertainties induced by LiDAR measurement noise, this paper introduces a LiDAR–inertial–visual odometry framework leveraging mergeable probabilistic voxel mapping. The method innovatively employs probabilistic voxel models to characterize uncertainties in environmental geometric plane features and optimizes computational efficiency through a voxel merging strategy. Additionally, it integrates color information from cameras to further enhance localization accuracy. Specifically, in the LiDAR–inertial odometry (LIO) subsystem, a probabilistic voxel plane model is constructed for LiDAR point clouds to explicitly represent measurement noise uncertainty, thereby improving the accuracy and robustness of point cloud registration. A voxel merging strategy based on the union-find algorithm is introduced to merge coplanar voxel planes, reducing computational load. In the visual–inertial odometry (VIO) subsystem, image tracking points are generated through a global map projection, and outlier points are eliminated using a random sample consensus algorithm based on a dynamic Bayesian network. Finally, state estimation accuracy is enhanced by jointly optimizing frame-to-frame reprojection errors and frame-to-map RGB color errors. Experimental results demonstrate that the proposed method achieves root mean square errors (RMSEs) of absolute trajectory error at 0.478 m and 0.185 m on the M2DGR and NTU-VIRAL datasets, respectively, while attaining real-time performance with an average processing time of 39.19 ms per-frame on the NTU-VIRAL datasets. Compared to state-of-the-art approaches, our method exhibits significant improvements in both accuracy and computational efficiency.

Keywords: LiDAR–inertial–visual odometry; voxel map; multi-sensor fusion; SLAM

Academic Editor: Christos J. Bouras

Received: 3 April 2025
Revised: 8 May 2025
Accepted: 22 May 2025
Published: 24 May 2025

Citation: Wang, B.; Bessaad, N.; Xu, H.; Zhu, X.; Li, H. Mergeable Probabilistic Voxel Mapping for LiDAR–Inertial–Visual Odometry. *Electronics* **2025**, *14*, 2142. https://doi.org/10.3390/electronics14112142

1. Introduction

Simultaneous localization and mapping (SLAM), as the foundational capability for environmental perception in intelligent unmanned systems, directly determines the decision-making competence of autonomous vehicles [1,2], mobile robots [3], and unmanned aerial vehicles [4] in complex scenarios through its localization accuracy and robustness. Current SLAM methodologies are categorized into two paradigms based on sensor modalities: visual SLAM [5–14], which establishes feature associations using visual sensors, and LiDAR-based SLAM [15–22], which leverages the geometric features of LiDAR point clouds.

Visual SLAM captures environmental data through camera sensors, extracting rich features such as textures and colors, which enables superior performance in feature-rich

environments. However, its mapping process relies on computationally intensive triangulation of multi-view image disparities to derive depth information at the pixel level. Additionally, depth estimation is affected by camera calibration errors and the baseline between multi-view images, which can lead to inaccuracies in depth estimation [23]. Furthermore, visual SLAM exhibits high sensitivity to illumination variations, with significant performance degradation under low-light conditions or in environments with strong specular reflections [24]. In contrast, LiDAR-based SLAM leverages laser scanners to provide high-precision depth measurements, ensuring stable operation across diverse lighting conditions and broader environmental applicability. Nevertheless, the inherent sparsity of LiDAR point clouds limits its capacity to discern textural details, particularly in geometrically impoverished environments such as tunnels and highways, where localization accuracy deteriorates markedly, and failure rates increase substantially [25].

To overcome the limitations of single-sensor systems in perception accuracy and environmental adaptability, recent research has proposed multi-sensor fusion strategies [26–34], aiming to achieve collaborative perception between visual systems and LiDAR through complementary advantages. Leveraging LiDAR's high-precision ranging and high-resolution 3D perception capabilities, it often serves as the primary source of prior environmental information in multi-sensor fusion localization systems. The point cloud maps constructed by tightly integrating LiDAR and inertial measurement unit (IMU) data not only provide initial pose estimation constraints for visual SLAM modules but also establish the spatial reference framework for global maps.

However, existing LiDAR–visual fusion methods typically treat LiDAR point clouds as deterministic data during map construction and employ the iterative closest point (ICP) algorithm [35] to register LiDAR points to predefined deterministic planes. This approach neglects the inherent noise characteristics in LiDAR measurements, potentially leading to reduced global map accuracy and, consequently, affecting the positioning precision of the entire fusion system. Therefore, LiDAR point clouds should be regarded as uncertain planes with probabilistic distributions rather than strict deterministic planes. Furthermore, current methodologies primarily confine the utilization of visual features to geometric constraints, failing to fully exploit the latent advantages embedded in RGB information.

In response to the aforementioned challenges, this paper proposes a LiDAR–inertial–visual odometry system based on mergeable probabilistic voxel mapping, with the following key contributions:

1. We propose a novel LiDAR–inertial–visual odometry (LIVO) framework that efficiently captures environmental geometric planar features and color information while simultaneously enhancing localization accuracy and maintaining real-time computational performance.
2. A point cloud processing method based on mergeable probabilistic voxel mapping is employed to probabilistically model the noise in the point clouds. By constructing voxel plane models through probabilistic modeling to estimate the system state, more accurate point cloud registration is achieved. Meanwhile, a hash table and union-find are used to merge voxel planes with coplanar relationships, effectively reducing the computational load.
3. In the visual–inertial subsystem (VIO), an accurate visual tracking and optimization scheme is designed. Tracking points are obtained through a global map projection, and outlier tracking points are removed using a random sample consensus algorithm based on a dynamic Bayesian network [36]. A joint optimization is then performed using frame-to-frame reprojection error and frame-to-map RGB color error, further improving system accuracy and robustness.

2. Related Work

This section will review previous research closely related to the proposed system, including LiDAR SLAM, visual SLAM, and LiDAR–visual SLAM.

2.1. Visual SLAM

Traditional visual SLAM methodologies can be categorized into feature-based methods and direct methods [37]. Feature-based visual SLAM estimates camera motion through the extraction and matching of visual features. For instance, MonoSLAM [5], as the pioneering monocular visual SLAM system, employs an extended Kalman filter (EKF) for system state estimation and utilizes Shi–Tomasi corner features for image tracking, achieving real-time recovery of 3D camera trajectories. PTAM [6] introduced a keyframe-based approach that selects representative frames for camera state estimation, thereby enabling computationally intensive yet more accurate bundle adjustment (BA) optimization. This framework innovatively decouples the pipeline into parallel tracking and mapping threads, establishing a paradigm shift in SLAM architecture design.

Building on this, ORB-SLAM2 [8] implements ORB feature points for data association and extends the multi-threaded paradigm through a tripartite architecture: tracking, local mapping, and loop closing. ORB-SLAM3 [9] introduces a tightly coupled visual–inertial SLAM framework based on nonlinear optimization, which jointly refines camera and IMU measurements within a sliding window to significantly enhance pose estimation accuracy. While integrating functionalities for monocular, stereo, and RGB-D cameras across visual, visual–inertial, and multi-map SLAM modules, its architectural complexity introduces substantial computational overhead. MSCKF [10] is a tightly coupled visual–inertial system based on EKF, which uses IMU for state prediction. It incorporates the IMU velocity, IMU measurement bias, and the multi-time camera poses in the sliding window into the state vector to perform 6-DOF motion estimation for visual–inertial odometry. VINS-Mono [11] strategically combines front-end optical flow tracking with back-end sliding window nonlinear optimization, achieving an operational equilibrium between real-time performance and localization precision.

In contrast to feature-based methods, direct methods bypass explicit feature extraction by minimizing photometric errors between consecutive frames to directly estimate camera states, thereby reducing computational complexity while enabling efficient tracking [37]. For instance, DTAM [12] estimates camera poses through alignment of entire images with dense depth maps, though its requirement for dense depth reconstruction from monocular inputs incurs significant computational demands. SVO [13], as a semi-direct visual odometry framework, acquires camera poses via direct matching of FAST corner patches within images, avoiding the exhaustive full image matching employed by pure direct methods. LSD-SLAM [14] presents a large-scale direct monocular SLAM approach that tracks camera motion through directly minimizing photometric errors and corrects accumulated errors through graph optimization. However, its effectiveness relies heavily on stringent photometric invariance assumptions that limit practical applicability.

Compared to feature-based methods, direct methods frequently demonstrate superior short-term tracking accuracy in low-texture environments [37], primarily due to their capacity to fully exploit pixel-level image information rather than relying exclusively on sparse salient feature points. Furthermore, by circumventing explicit feature extraction stages, direct methods inherently achieve superior computational efficiency. Capitalizing on these merits, our framework strategically implements direct methods within the VIO subsystem for tracking and matching, which helps to improve the overall robustness and localization precision.

2.2. LiDAR SLAM

LiDAR SLAM achieves pose estimation between consecutive frames through scan-matching algorithms. LOAM [15], as a classic LiDAR odometry method, implements localization based on KD-Tree [38] scan matching of edge and plane feature points. However, LOAM employs a loosely coupled architecture that omits the integration of IMU and LiDAR measurements in joint optimization. In contrast, LIO-SAM [16] implements a tightly coupled smoothing and mapping framework, fusing IMU pre-integration, LiDAR odometry, loop closure factors, and GPS measurements within a unified factor graph to achieve enhanced localization precision. FAST-LIO [17] establishes a tightly coupled fusion of IMU and LiDAR data through an error-state iterative Kalman filter (ESIKF) and manages the point cloud map using KD-Tree. It also proposes a new Kalman gain computation formula, where the computational cost depends on the state dimension rather than the measurement dimension, significantly reducing the computational load and improving processing speed.

However, KD-Tree-based methods require retrieving at least three nearest neighbors for each point-to-plane correspondence to fit the corresponding plane. As the map grows, the cost of rebuilding the KD-Tree also increases. To improve scan-matching efficiency, FAST-LIO2 [18] develops an incremental iKD-Tree [39] to effectively organize the point cloud map. This structure allows for efficient nearest-neighbor searches, further reducing the computational load while making the map organization more efficient. Building on this, Faster-LIO [19] proposes a tightly coupled LiDAR odometry method based on incremental voxel mapping. Compared to iKD-Tree, its map update and nearest-neighbor search speeds are further enhanced. Voxelmap [20] and Voxelmap++ [21] organize and manage maps using octrees and union-find structures, respectively. They also perform probabilistic voxel plane modeling of LiDAR points, eliminating the reliance on time-consuming nearest-neighbor searches, thus improving computational speed. Voxel-SLAM [22] proposes a voxel-based tightly coupled LiDAR–inertial SLAM system that unifies environmental information representation through adaptive voxel maps and utilizes short-term, mid-term, and long-term multi-scale data association for high-precision localization and mapping.

The LIO subsystem in this paper is primarily based on Voxelmap++ [21], which explicitly parameterizes the noise in LiDAR point measurements using voxel modeling, achieving a balance between high precision and real-time performance. However, the merging of voxel planes introduces cumulative errors and neglects the differences between voxel planes. To address this, this paper utilizes image information from the VIO subsystem to further optimize the system state, effectively compensating for the limitations of the LIO subsystem.

2.3. LiDAR–Visual SLAM

To enhance the accuracy and robustness of SLAM systems, fusing data from LiDAR, cameras, and IMUs has proven to be an effective strategy. The LIC-Fusion series [26,27] employs the MSCKF framework [10] to tightly couple LiDAR features, IMU measurements, and sparse visual feature points, utilizing a sliding window approach for feature tracking. LVI-SAM [28] leverages a factor graph to tightly couple visual–inertial and LiDAR–inertial subsystems, incorporating loop closure detection to achieve robust and high-precision state estimation and mapping. FAST-LIVO [29] integrates FAST-LIO2 [18] and SVO [13], utilizing a LiDAR–inertial subsystem to generate global point clouds while attaching image patches to each map point. It optimizes state estimation by minimizing photometric errors and introduces a novel outlier rejection method to enhance system accuracy and robustness. Building upon this foundation, FAST-LIVO2 [30] further enhances system performance by efficiently fusing IMU, LiDAR, and visual data through the ESIKF. It addresses dimensional

mismatch in heterogeneous data via a sequential update strategy. Additionally, FAST-LIVO2 processes visual and LiDAR data using direct methods, manages map points and high-resolution image measurements through a unified voxel map, dynamically updates reference image patches, and estimates exposure time in real time to improve system robustness and accuracy.

R2LIVE [31] introduces a hybrid framework combining high-frequency ESIKF-based odometry with low-frequency factor graph optimization to further improve overall accuracy through factor graph optimization. Building upon this framework, R3LIVE [32] employs a LiDAR–inertial subsystem to construct global geometric structures while utilizing the visual–inertial subsystem to optimize state estimates through RGB photometric constraints, enabling real-time generation of RGB point cloud maps. R3LIVE++ [33] further modifies R3LIVE by utilizing a radiance map, optimizing the state based on the invariance of radiance values, which further improves localization and mapping accuracy. LVIO-Fusion [34] implements dynamic voxel mapping in its LIO subsystem and proposes a coarse-to-fine hierarchical state estimation approach in the visual–inertial odometry (VIO) subsystem, as well as using a factor-graph-based joint optimization of multi-sensor constraints.

However, existing LiDAR–visual fusion approaches do not fully consider the uncertainty caused by noise in LiDAR measurements when constructing the point cloud map. To address this, this paper explicitly parameterizes and models the measurement noise of LiDAR points using a mergeable probabilistic voxel model and optimizes voxel map management. Additionally, by jointly optimizing the measurements from IMU, LiDAR, and cameras, the paper achieves a collaborative improvement in both localization accuracy and system robustness.

3. Notation and System Overview

3.1. Notation

The state vector $\boldsymbol{x} \in \mathbb{R}^{29}$ of the proposed method is defined as follows:

$$\boldsymbol{x} = \left[{}^{G}\boldsymbol{R}_I^T, {}^{G}\boldsymbol{p}_I^T, {}^{G}\boldsymbol{v}^T, \boldsymbol{b}_a^T, \boldsymbol{b}_g^T, {}^{G}\boldsymbol{g}^T, {}^{G}\boldsymbol{R}_C^T, {}^{G}\boldsymbol{p}_C^T, {}^{I}t_C, f_x, f_y, c_x, c_y \right]^T \tag{1}$$

where $({}^{G}\boldsymbol{R}_I, {}^{G}\boldsymbol{p}_I)$ represents the rotation matrix (unitless) and translation vector (unit: m) from the IMU frame to the global frame, ${}^{G}\boldsymbol{v}$ is the IMU velocity (unit: m/s) in the global frame, $(\boldsymbol{b}_a, \boldsymbol{b}_g)$ are the IMU acceleration bias (unit: m/s^2) and gyroscope bias (unit: rad/s), and ${}^{G}\boldsymbol{g}$ is the gravity acceleration (unit: m/s^2) in the global frame. $({}^{G}\boldsymbol{R}_C, {}^{G}\boldsymbol{p}_C)$ denote the rotation matrix and translation vector from the camera frame to the global frame, ${}^{I}t_C$ is the time offset (unit: s) between the camera and IMU, and (f_x, f_y, c_x, c_y) are the intrinsic parameters of the camera (unit: pixels).

Additionally, there are some special symbols in this paper: $\boxminus$ and $\boxplus$ represent "boxplus" and "boxminus", encapsulating manifold operations. $\sum_{(\cdot)}$ is the covariance matrix of the vector $(\cdot)$. $\hat{\mathbf{x}}$ is the prior estimate of the state vector. $\lfloor(\cdot)\rfloor_\wedge$ represents the skew-symmetric matrix of $(\cdot)$.

3.2. System Overview

Unlike feature-based LIVO systems, this paper does not directly extract feature points in either the LIO subsystem or the VIO subsystem. The system framework is shown in Figure 1. The LIO subsystem constructs a point cloud map using a mergeable probabilistic voxel plane model, integrating LiDAR point clouds and IMU information to obtain a rough estimate of the state vector. The VIO subsystem uses the prior information provided by the LIO subsystem, first removing abnormal tracking points and then further refining the estimation of the state vector.

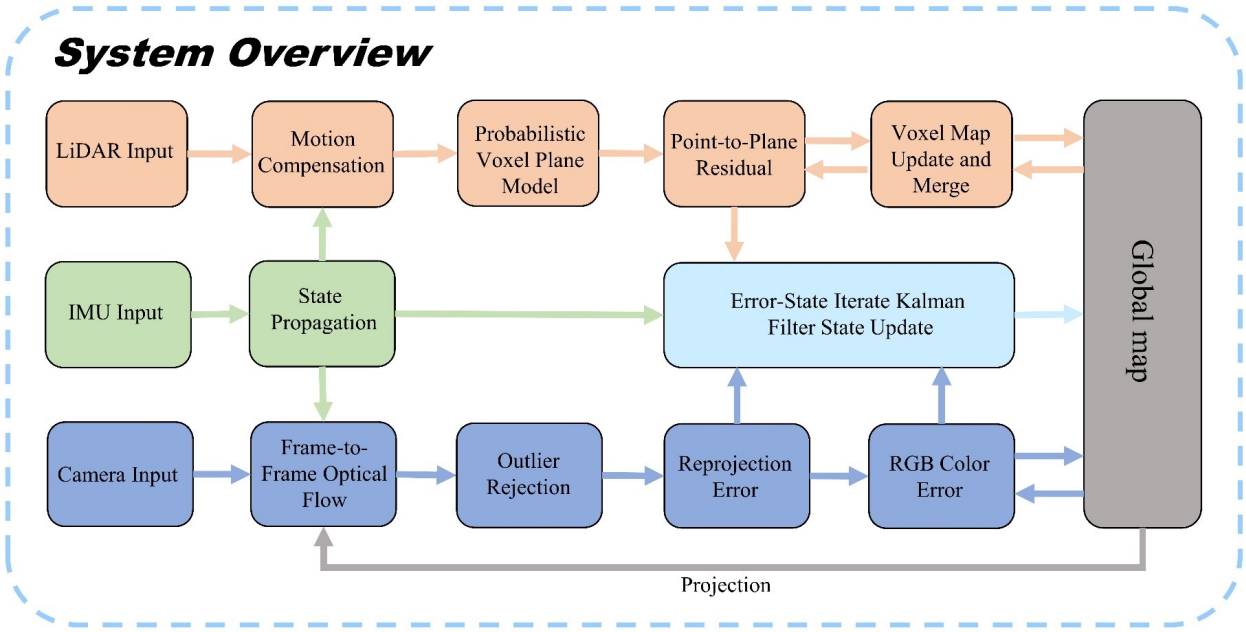


Figure 1. System framework. The orange, green, and blue modules correspond to the LiDAR, IMU, and camera data processing pipelines, respectively; the gray module represents global map construction, while the light-blue module demonstrates the state vector update process based on ESIKF.

Specifically, in the LIO subsystem, IMU pre-integration is used to perform motion compensation on the point cloud to remove motion distortion. The corrected point cloud is then divided into fixed-size voxel grids, and probabilistic voxel planes are constructed by modeling point cloud uncertainties. The state vector is coarsely estimated by minimizing point-to-plane residuals. Hash tables and union-find sets are used to merge voxel planes with coplanar relationships to update and build the voxel map. Finally, the LiDAR point cloud identified as part of the voxel plane is added to the global map.

In the VIO subsystem, the state output from the LIO subsystem is used as the initial value for the VIO subsystem. The depth information from point clouds in the LIO subsystem is directly used as depth information for images in VIO. The global map is projected onto the image frames to directly acquire tracking points, and then the Lucas–Kanade (LK) optical flow method is employed to track these projected points. Additionally, a random sample consensus algorithm based on a dynamic Bayesian network [36] is used to further eliminate outliers. The state vector is further refined by minimizing both frame-to-frame reprojection errors and frame-to-map RGB photometric errors, ultimately maintaining and updating the global RGB map.

4. LiDAR–Inertial Odometry Subsystem

In the LIO subsystem, motion distortion correction is first applied to incoming LiDAR frames using a backward propagation algorithm [17]. Subsequently, based on the error-state iterative Kalman filter (ESIKF) framework, system state estimation is achieved by constructing point-to-plane residual constraints based on 3DoF probabilistic planar models in the voxel map (see Section 4.3). The 3DoF probabilistic plane model is detailed in Section 4.1. For voxel map construction, the LIO subsystem projects each point of the point cloud into the corresponding voxel and builds or updates the voxel map using union-find sets and hash tables. During the voxel map update, it is considered that many small coplanar voxel planes may be sub-planes under a larger parent plane. These sub-planes are merged using the union-find set to re-estimate the parent plane parameters (see Section 4.2). Finally, the LiDAR point cloud identified as part of the plane voxel is added to the global

map, allowing the VIO subsystem to extract depth information for real-world 3D objects from the LiDAR data.

4.1. 3DOF Probabilistic Plane Representation

Since the parameters of a plane are estimated using points on the plane, any noise in the points will increase the uncertainty in the plane estimation. Traditional point cloud map construction methods directly use raw sensor observation data and fail to effectively quantify the impact of LiDAR point noise on the uncertainty of plane parameter estimation. To address this, this paper considers two factors that contribute to the uncertainty of LiDAR points in the voxel map: one originates from the raw point cloud measurement noise in the local LiDAR coordinate system, which is relative to the local LiDAR frame; the other arises from the LiDAR pose estimation error during the projection of the point cloud data from the local coordinate system to the global world coordinate system. It is important to note that the noise distribution of each point within the plane has a cumulative effect on the accuracy of plane parameter estimation. Based on the above analysis, this paper sequentially constructs uncertainty models for both points and planes.

4.1.1. Uncertainty of the Point

According to the analysis of LiDAR sensor measurement noise in [40], the uncertainty of LiDAR points includes both distance uncertainty and orientation uncertainty, as shown in Figure 2b. $\boldsymbol{\omega}_i$ represents the tangent plane where the LiDAR point resides, and $\boldsymbol{\delta}_{\boldsymbol{\omega}_i} \sim N(\mathbf{0}_{2\times1}, \boldsymbol{\Sigma}_{\boldsymbol{\omega}_i})$ represents the noise of the LiDAR point along the tangent plane. d_i denotes the depth of the LiDAR point, and $\delta_{d_i} \sim N(0, \Sigma_{d_i})$ represents the range noise. Therefore, the noise ${}^{L}P_i$ of the LiDAR point $\boldsymbol{\delta}_{L_{\boldsymbol{P}_i}}$ and its covariance $\boldsymbol{\Sigma}_{L_{\boldsymbol{P}_i}}$ are calculated as follows:

$$\boldsymbol{\delta}_{L_{\boldsymbol{P}_i}} = \begin{bmatrix} \boldsymbol{\omega}_i & -d_i \lfloor \boldsymbol{\omega}_i \rfloor_\wedge \boldsymbol{N}(\boldsymbol{\omega}_i) \end{bmatrix} \begin{bmatrix} \delta_{d_i} \\ \boldsymbol{\delta}_{\boldsymbol{\omega}_i} \end{bmatrix} \tag{2}$$

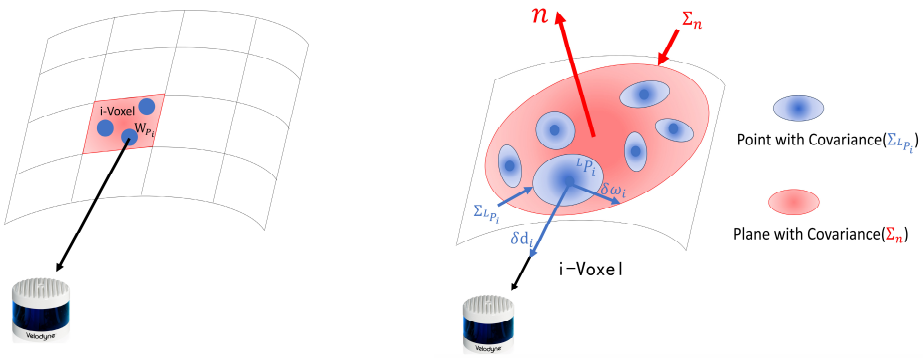
$$\boldsymbol{\Sigma}_{L_{\boldsymbol{P}_i}} = \boldsymbol{A}_i \begin{bmatrix} \Sigma_{d_i} & \mathbf{0}_{1\times2} \\ \mathbf{0}_{2\times1} & \boldsymbol{\Sigma}_{\boldsymbol{\omega}_i} \end{bmatrix} \boldsymbol{A}_i^T \tag{3}$$

where $\boldsymbol{N}(\boldsymbol{\omega}_i) = \begin{bmatrix} \boldsymbol{N}_1 & \boldsymbol{N}_2 \end{bmatrix}$ is the standard orthonormal basis at the tangent plane $\boldsymbol{\omega}_i$, and $\lfloor \rfloor_\wedge$ denotes the skew-symmetric matrix. A detailed derivation of Equation (2) can be found in [40]. By using the pose estimation results $({}^{G}\boldsymbol{R}_I, {}^{G}\boldsymbol{p}_I)$ from IMU pre-integration, the LiDAR point ${}^{L}\boldsymbol{P}_i$ is further projected into the world coordinate system, as shown in Equation (4). Thus, the uncertainty of the LiDAR point ${}^{W}\boldsymbol{P}_i$ can be represented as shown in Equation (5).

$${}^{W}\boldsymbol{P}_i = {}^{G}\boldsymbol{R}_I \, {}^{L}\boldsymbol{P}_i + {}^{G}\boldsymbol{p}_I \tag{4}$$

$$\boldsymbol{\Sigma}_{{}^{W}\boldsymbol{P}_i} = {}^{G}\boldsymbol{R}_I \boldsymbol{\Sigma}_{{}^{L}\boldsymbol{P}_i} {}^{G}\boldsymbol{R}_I{}^{T} + {}^{G}\boldsymbol{R}_I \left\lfloor {}^{L}\boldsymbol{P}_i \right\rfloor_\wedge \boldsymbol{\Sigma}_{{}^{G}\boldsymbol{R}_I} \left\lfloor {}^{L}\boldsymbol{P}_i \right\rfloor_\wedge^{T} {}^{G}\boldsymbol{R}_I{}^{T} + \boldsymbol{\Sigma}_{{}^{G}\boldsymbol{p}_I} \tag{5}$$

where $\boldsymbol{\Sigma}_{{}^{G}\boldsymbol{R}_I}$ and $\boldsymbol{\Sigma}_{{}^{G}\boldsymbol{p}_I}$ are the uncertainties of ${}^{G}\boldsymbol{R}_I$ and ${}^{G}\boldsymbol{R}_I$ at the tangent plane, with more details available in [20]. The uncertainty of the point is the cornerstone of the plane's uncertainty, and the plane's uncertainty is determined by the uncertainty of the points within the plane.



(**a**) (**b**)

Figure 2. (**a**) illustrates the representation of LiDAR points on a voxel plane, where *i*-Voxel represents the *i*-th voxel. (**b**) depicts the uncertainty model for LiDAR points and the plane. The blue part represents the LiDAR points and their parameters, while the red part represents the plane and its parameters.

4.1.2. Uncertainty of a 3DoF Plane

For a plane composed of N LiDAR points ${}^{W}\boldsymbol{P}_i (i = 1, \ldots, N)$, the uncertainty of each point is represented by its covariance matrix $\Sigma_{W_{P_i}}$. Based on the least squares principle, planar uncertainty can be computed via a dimensionality-reduced parameterization method [41]. Specifically, the plane equation can be normalized along a principal axis. When the z-axis is chosen as the principal axis, the normalized plane representation utilizes three parameters, $\boldsymbol{n} = [a, b, d]^T$, to define the plane, rather than the six parameters comprising the normal vector and the plane's center point. This approach offers a three-degree-of-freedom (3DoF) representation, reducing memory usage compared to the six-parameter method. However, this representation becomes singular when the z-component of the plane's normal approaches zero. This issue can be addressed by projecting the point cloud onto the coordinate axes. All points ${}^{W}\boldsymbol{P}_i$ on the plane satisfy Equation (6), from which Equations (7) and (8) are derived. A closed-form solution for the plane parameters n (Equation (9)) is obtained via least squares optimization, ultimately yielding the planar uncertainty estimation model in Equation (10).

$$ax + by + z + d = 0 \tag{6}$$

$$\begin{bmatrix} x_1 & y_1 & 1 \\ x_2 & y_2 & 1 \\ \vdots & \vdots & \vdots \\ x_n & y_n & 1 \end{bmatrix} \boldsymbol{n} = \begin{bmatrix} -z_1 \\ -z_2 \\ \vdots \\ -z_n \end{bmatrix} \tag{7}$$

$$\boldsymbol{n} = \frac{\boldsymbol{A}^*}{|\boldsymbol{A}|}\boldsymbol{e} \tag{8}$$

where $\boldsymbol{A}^*$ is the adjugate matrix of $\boldsymbol{A}$, and $\boldsymbol{A}$ and $\boldsymbol{e}$ can be expressed as follows:

$$\begin{aligned} \boldsymbol{A} &= \begin{bmatrix} \sum x_i x_i & \sum x_i y_i & \sum x_i \\ \sum x_i y_i & \sum y_i y_i & \sum y_i \\ \sum x_i & \sum y_i & N \end{bmatrix} \\ \boldsymbol{e} &= \begin{bmatrix} -\sum x_i z_i & -\sum y_i z_i & -\sum z_i \end{bmatrix}^T \end{aligned} \tag{9}$$

$$\boldsymbol{\Sigma}_{\boldsymbol{n}} = \sum_{i}^{N} \frac{\partial \boldsymbol{n}}{\partial {}^{W}\boldsymbol{P}_i} \Sigma_{{}^{W}\boldsymbol{P}_i} \frac{\partial \boldsymbol{n}}{\partial {}^{W}\boldsymbol{P}_i}^T \tag{10}$$

Note that the elements recorded in $\boldsymbol{A}$, $\boldsymbol{A}^*$, and $\boldsymbol{e}$ are the summation results of all points. Therefore, in Section 4.2 on voxel construction, the 3DoF plane representation can be updated incrementally, effectively improving the efficiency of voxel map updates.

4.2. Voxel Map Construction and Updates

Dividing the voxel planes with completed probabilistic modeling into independent units indeed enhances the system's robustness. However, directly storing each independent voxel, as the LiDAR point clouds are divided into fixed-size voxels (e.g., $0.5 \times 0.5 \times 0.5$ m), would lead to a significant increase in storage overhead, which is clearly impractical in engineering applications. Moreover, existing methods do not fully account for the inherent correlations between voxel planes. That is, when facing large planar scenes, it is important to consider whether small voxel planes in the voxel map are coplanar. For example, in large continuous planar scenes (e.g., walls), a single continuous plane is often segmented into multiple sub-planes, which should inherently belong to the same parent plane. This means that sub-planes $\mathcal{P}^k$ with a coplanar relationship share a common larger parent plane $\mathcal{P}^f$. The 3DoF estimation results of sub-plane $\mathcal{P}^k$ can be treated as a covariance measurement of the parent plane $\mathcal{P}^f$. To address this, this paper uses the union-find algorithm to merge these sub-planes $\mathcal{P}^k$ in order to obtain a more accurate 3DoF estimate of the parent plane $\mathcal{P}^f$. This approach effectively saves memory usage and improves the accuracy of plane fitting.

4.2.1. Voxel Map Construction

This paper constructs voxel maps through a discrete spatial partitioning approach based on voxelized grids. Specifically, raw LiDAR point clouds are discretized into fixed-size voxel grids. A hash table enables rapid spatial indexing, while a union-find data structure maintains topological relationships between voxels, thereby constructing a dynamically updatable voxel map. For the initial LiDAR frame, points are spatially mapped and filled into the corresponding voxel units (see Figure 2a). For the filled voxels, the plane parameters $\boldsymbol{n} = [a, b, d]^T$ and their uncertainty $\boldsymbol{\Sigma_n}$ are computed based on Equations (8) and (10). Through eigenvalue analysis, when the minimum eigenvalue of $\boldsymbol{\Sigma_n}$ satisfies $\lambda_{min} < \tau$ (with a threshold $\tau = 0.01$), the voxel is deemed a valid plane voxel, and its geometric parameters are retained for subsequent processing.

For subsequent LiDAR frame processing, newly acquired point clouds are dynamically registered into the existing voxel map. Specifically, when it is detected that a voxel has already been established at the corresponding location in the voxel map and the voxel has not yet converged (defined as the number of points within the voxel being less than 50), the system incorporates the current point cloud into the target voxel and incrementally updates the plane parameters n and uncertainty covariance matrix $\boldsymbol{\Sigma_n}$ of the voxel. If a voxel has not been established at the corresponding location, a new voxel is initialized, and the plane parameters are calculated. To conserve system memory, the system sets a maximum capacity limit of 50 points of cloud data for a single voxel. Once the voxel capacity is saturated, updates to the voxel's plane parameters are halted, and the voxel is marked as a converged voxel (based on the conclusion in reference [20], the uncertainty of the plane parameters converges when the number of points reaches 50). At the same time, the voxel storage is cleared to release memory resources, effectively balancing computational efficiency and storage requirements.

4.2.2. Voxel Merging and Updating

After probabilistic modeling of the voxels and constructing probabilistic voxel planes, this paper considers the inherent spatial correlations between voxel planes and merges those with coplanar relationships. Specifically, for a converged voxel plane $\mathcal{P}^n$, the system searches for other converged voxel planes $\mathcal{P}_i^k (i = 1, \ldots, n)$ that are coplanar with $\mathcal{P}^n$ and

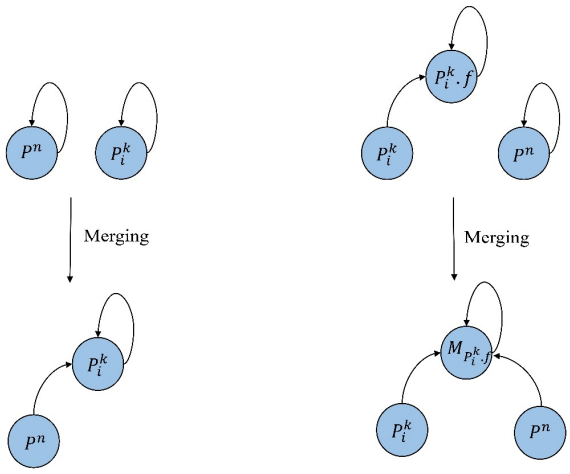
merges them. By introducing the Mahalanobis distance metric (as shown in Equation (11)), the similarity between planes is quantified. When the computed result is below the critical threshold determined by the χ^2 distribution at the 95% confidence level, it is determined that the geometric coplanar condition is satisfied. At this point, voxels $\mathcal{P}^n$ and $\mathcal{P}_i^k$ are merged to obtain ${}^M\mathcal{P}_i^k \cdot f$. This fusion mechanism effectively reduces the system's memory usage while maintaining the geometric consistency of the 3D reconstruction.

$$d = \left(\boldsymbol{n}_{\mathcal{P}_i^k} - \boldsymbol{n}_{\mathcal{P}^n}\right)\left(\boldsymbol{\Sigma}_{n_{\mathcal{P}_i^k}} + \boldsymbol{\Sigma}_{n_{\mathcal{P}^n}}\right)^{-1}\left(\boldsymbol{n}_{\mathcal{P}_i^k} - \boldsymbol{n}_{\mathcal{P}^n}\right)^T \tag{11}$$

$$\boldsymbol{n}_{M\mathcal{P}_i^k \cdot f} = \frac{\left\|\boldsymbol{\Sigma}_{\boldsymbol{n}_{\mathcal{P}^n}}\right\|_2 \boldsymbol{n}_{\mathcal{P}_i^k} + \left\|\boldsymbol{\Sigma}_{n_{\mathcal{P}_i^k}}\right\|_2 \boldsymbol{n}_{\mathcal{P}^n}}{\left\|\boldsymbol{\Sigma}_{\boldsymbol{n}_{\mathcal{P}^n}}\right\|_2 + \left\|\boldsymbol{\Sigma}_{n_{\mathcal{P}_i^k}}\right\|_2} \tag{12}$$

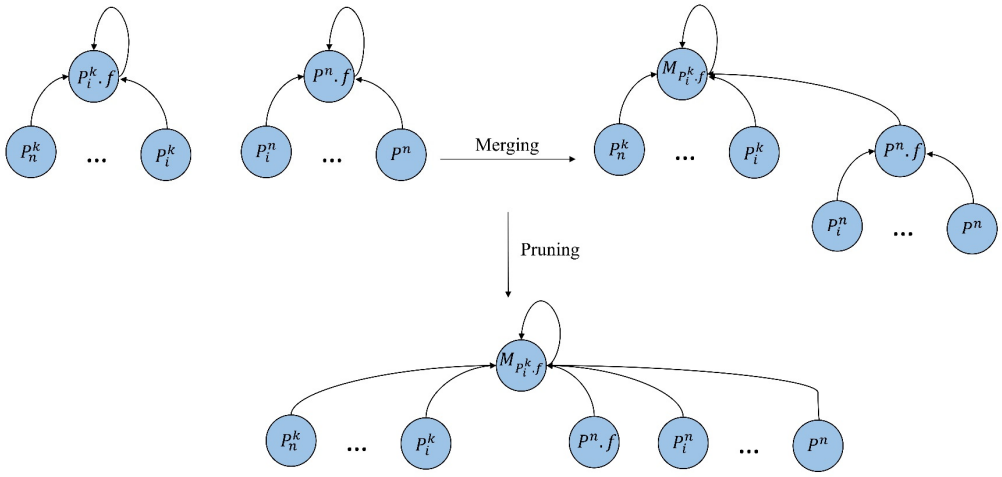
$$\boldsymbol{\Sigma}_{\boldsymbol{n}_{M\mathcal{P}_i^k \cdot f}} = \frac{\left(\left\|\boldsymbol{\Sigma}_{\boldsymbol{n}_{\mathcal{P}^n}}\right\|_2\right)^2 \boldsymbol{\Sigma}_{n_{\mathcal{P}_i^k}} + \left(\left\|\boldsymbol{\Sigma}_{n_{\mathcal{P}_i^k}}\right\|_2\right)^2 \boldsymbol{\Sigma}_{n_{\mathcal{P}^n}}}{\left(\left\|\boldsymbol{\Sigma}_{\boldsymbol{n}_{\mathcal{P}^n}}\right\|_2 + \left\|\boldsymbol{\Sigma}_{n_{\mathcal{P}_i^k}}\right\|_2\right)^2} \tag{13}$$

where $\| \ \|_2$ denotes the Euclidean norm. For merged voxel planes ${}^M\mathcal{P}_i^k \cdot f$, Equations (12) and (13) are used to obtain the corresponding plane's covariance, $\boldsymbol{\Sigma}_{\boldsymbol{n}_{M\mathcal{P}_i^k \cdot f}}$, and the 3DoF plane parameter representation, $\boldsymbol{n}_{M\mathcal{P}_i^k \cdot f}$. These equations are based on a simple weighted averaging algorithm with minimal trace. As shown in Figure 3, there are two typical scenarios for plane merging. In Figure 3a, a simple merging case is illustrated, where the number of points is small, and the parent plane of the current plane is directly set to the parent plane of the neighboring plane. In Figure 3b, a more complex merging scenario is shown. When the converged plane $\mathcal{P}^n$ and its adjacent converged plane $\mathcal{P}_i^k$ are merged, the parent plane of the current plane $\mathcal{P}^n \cdot f$ is set to point to the parent plane of the neighboring plane $\mathcal{P}_i^k \cdot f$, and the merged plane undergoes pruning, ensuring that the union-find height remains below 2.



(**a**)

Figure 3. *Cont.*



(**b**)

Figure 3. Illustration of the scenarios of plane merging. (**a**) shows a simple case of plane merging. (**b**) depicts a complex case of plane merging.

4.3. State Estimation Based on ESIKF

In the *i*-th point-to-plane matching, the observation equation for the LiDAR point-to-plane distance is given as follows:

$$h(\hat{x}_k, n_i) = \frac{\Omega^T \left({}^G R_I \, {}^L P_i + {}^G p_I \right) + d}{\|\Omega\|} \tag{14}$$

where Ω is the normalized normal vector of the plane ${}^M\mathcal{P}_i^k \cdot f$, represented as $[a, b, 1]^T$. Combining the prior state estimate $\hat{x}_k$ and its covariance $\mathbf{\Sigma}_{\delta\hat{\mathbf{x}}_k}$ obtained from the IMU state propagation, the prior information is integrated with the point-to-plane distance matching observation to construct the maximum a posteriori (MAP) model as shown in Equation (15). This model consists of two components: the prior state term and the observation term. The solution process is implemented based on an error-state iterative Kalman filter (ESIKF).

$$\min_{\delta\check{\mathbf{x}}_k} \left\{ \|\check{\boldsymbol{x}}_k \boxminus \hat{\boldsymbol{x}}_k + \boldsymbol{J}^k \delta\check{\boldsymbol{x}}_k\|^2_{\mathbf{\Sigma}_{\delta\hat{\mathbf{x}}_k}} + \sum_{i=1}^{N} \|h(\check{\boldsymbol{x}}_k, {}^L\boldsymbol{P}_i) + \boldsymbol{H}_i^h \delta\check{\boldsymbol{x}}_k\|^2_{\sum_{l_i}} \right\} \tag{15}$$

where $\|x\|^2_{\sum} = x^T \sum^{-1} x$ represents the Mahalanobis distance with respect to the covariance matrix $\sum$, $\check{\mathbf{x}}_\mathbf{k}$ denotes the posterior state estimate, and J^k denotes the Jacobian matrix that projects state errors from the tangent space of the prior state estimate $\hat{\mathbf{x}}_k$ to the tangent space of $\check{\mathbf{x}}_\mathbf{k}$ (detailed computation can be found in [17]). H_i^h is the Jacobian matrix of the observation equation h, and $\sum_{l_i}$ is the observation noise covariance matrix. The calculation methods for these are as follows:

$$\boldsymbol{H}_i^h = \frac{\partial h\left(\check{\boldsymbol{x}}_k \boxplus \delta\check{\boldsymbol{x}}_k, {}^L\boldsymbol{P}_i\right)}{\partial \boldsymbol{\delta}_{\check{\mathbf{x}}_k}} \tag{16}$$

$$
\begin{aligned}
\sum_{l_i} &= \boldsymbol{J}_{v_i} \boldsymbol{\Sigma}_{\boldsymbol{n}_{M_{\mathcal{P}_i^k \cdot f}}, {}^L\boldsymbol{P}_i} \boldsymbol{J}_{v_i}^T \\
\boldsymbol{\Sigma}_{\boldsymbol{n}_{M_{\mathcal{P}_i^k \cdot f}}, {}^L\boldsymbol{P}_i} &= \begin{bmatrix} \boldsymbol{\Sigma}_{\boldsymbol{n}_{M_{\mathcal{P}_i^k \cdot f}}} & \boldsymbol{0}_{3\times 3} \\ \boldsymbol{0}_{3\times 3} & \boldsymbol{\Sigma}_{{}^L\boldsymbol{P}_i} \end{bmatrix} \\
\boldsymbol{J}_{v_i} &= \begin{bmatrix} \boldsymbol{J}_{n_i}, \boldsymbol{J}_{{}^L P_i} \end{bmatrix} \\
\boldsymbol{J}_{n_i} &= \frac{1}{\|\boldsymbol{\Omega}\|} \begin{bmatrix} x_i \left(1 - \frac{1}{\|\boldsymbol{\Omega}\|^2} h_i(\boldsymbol{x}_i, 0)\right) \\ y_i \left(1 - \frac{1}{\|\boldsymbol{\Omega}\|^2} h_i(\boldsymbol{x}_i, 0)\right) \\ 1 \end{bmatrix} \\
\boldsymbol{J}_{{}^L P_i} &= \frac{1}{\|\boldsymbol{\Omega}\|} \boldsymbol{\Omega}^T \times {}^G\boldsymbol{R}_I
\end{aligned}
\tag{17}
$$

where $\Sigma_{n_{M_{\mathcal{P}_i^k \cdot f}}}$ is the observation error covariance of the converged plane, and $\Sigma_{{}^L P_i}$ is the observation error covariance of the point. The method for calculating the Kalman gain in reference to [17] is as follows:

$$
\boldsymbol{K} = \left(\boldsymbol{H}^T \boldsymbol{R}^{-1} \boldsymbol{H} + \boldsymbol{P}^{-1}\right)^{-1} \boldsymbol{H}^T \boldsymbol{R}^{-1} \tag{18}
$$

$$
\boldsymbol{H} = \left[\boldsymbol{H}_1^{h\,T}, \ldots, \boldsymbol{H}_m^{h\,T}\right]^T, \quad \boldsymbol{R} = diag\left(\sum_{l_1}, \ldots, \sum_{l_m}\right) \tag{19}
$$

$$
\boldsymbol{P} = \left(\boldsymbol{J}^k\right)^{-1} \boldsymbol{\Sigma}_{\delta \hat{\boldsymbol{x}}_k} \left(\boldsymbol{J}^k\right)^{-T} \tag{20}
$$

The state vector $\check{\boldsymbol{x}}_k$ can be updated by the following equation, where $\check{\boldsymbol{z}}_k$ is the observation residual.

$$
\check{\boldsymbol{x}}_k = \check{\boldsymbol{x}}_k \boxplus \left(-K\check{\boldsymbol{z}}_k - (\boldsymbol{I} - \boldsymbol{K}\boldsymbol{H})\left(\boldsymbol{J}^k\right)^{-1} (\check{\boldsymbol{x}}_k \boxminus \hat{\boldsymbol{x}}_k)\right) \tag{21}
$$

$$
\check{\boldsymbol{z}}_k = \left[h\left(\hat{\boldsymbol{x}}_k, {}^L\boldsymbol{P}_1\right), \ldots, h\left(\hat{\boldsymbol{x}}_k, {}^L\boldsymbol{P}_m\right)\right]^T \tag{22}
$$

The above state vector update process is carried out over multiple iterations and terminates as soon as the change in the state falls below a predefined threshold or the maximum number of iterations is reached. At that point, the system deems the current state vector to be a reliable estimate and updates the prior state and its covariance matrix according to the standard Kalman filter update equations. Finally, the refined state is used as the new initial condition for IMU propagation in the LIO or VIO subsystem, ensuring that subsequent optimizations start from the most accurate available estimate.

$$
\hat{\boldsymbol{x}}_{\boldsymbol{k}} = \check{\boldsymbol{x}}_{\boldsymbol{k}}, \quad \hat{\boldsymbol{\Sigma}}_{\delta \hat{\boldsymbol{x}}_k} = (\boldsymbol{I} - \boldsymbol{K}\boldsymbol{H}) \boldsymbol{\Sigma}_{\delta \hat{\boldsymbol{x}}_k} \tag{23}
$$

5. Visual–Inertial Odometry Subsystem

In the visual–inertial odometry (VIO) system, an initial state estimate is obtained via IMU pre-integration, and points from the global map are projected onto the current image plane to establish spatial correspondences between map points and image pixels, with pixel depth information provided by the LiDAR. Subsequently, the Lucas–Kanade (LK) optical flow method is employed for real-time feature tracking across consecutive image frames. To mitigate the impact of outliers on system accuracy, a random sample consensus (RANSAC) algorithm based on a dynamic Bayesian network [36]—an enhanced variant of the traditional RANSAC framework—is introduced to dynamically assess the reliability of sampled data subsets through probabilistic modeling and reject erroneous feature points. The system further refines state estimates by optimizing frame-to-frame reprojection errors using the ESIKF. Additionally, RGB photometric constraints are constructed by extracting

image color information, and multi-modal fusion optimization is achieved through joint minimization of RGB color errors using the ESIKF framework, thereby enhancing both localization precision and environmental representation consistency.

5.1. Outlier Rejection

Following LK optical flow tracking, a random sample consensus algorithm based on a dynamic Bayesian network [36] is employed to eliminate outliers. Specifically, this algorithm updates the inlier scores of individual data points during each iteration using the dynamic Bayesian network. Weighted sampling is conducted at each iteration based on these updated scores, and the algorithm's termination criteria are dynamically adjusted according to the inlier and outlier probabilities of the data points. Compared to traditional RANSAC methods, this approach offers enhanced computational precision and reduced processing time. An overview of the algorithm is presented in Algorithm 1.

Algorithm 1: BANSAC algorithm outline

Input: Data Q and number of iterations K
Output: Best model θ* and C*

1. $k \longleftarrow 1$.
2. **while** $k < K$ **do**
3. $\quad S^k \longleftarrow$ weighted_sampling(Q, P^{k-1})
4. $\quad \theta^k, O^k \longleftarrow$ hypothesis(S^k)
5. $\quad C^k \longleftarrow$ model_evaluatiom(Q, θ^k, O^k)
6. $\quad \theta^*, O^*, C^* \longleftarrow$ best_model(θ^k, O^k, C^k)
7. $\quad P^k \longleftarrow$ update_probabilities(θ^k, O^k, C^k)
8. $\quad \tilde{O}^k \longleftarrow$ number_of$\left(p_i^k < \tau\right)$
9. $\quad$ **if** $\tilde{O}^k \geqq O^*$ **do**
10. $\quad\quad$ **break**
11. $\quad$ **endif**
12. $\quad k \longleftarrow k + 1$
13. **end while**

The algorithm takes as input a dataset $\boldsymbol{Q} = \{x_1, \ldots, x_N\}$ and outputs a hypothesized model θ^* along with the classification C^* of the data points in Q as inliers or outliers based on the hypothesis model. In the k-th iteration, the inlier probabilities $P^{k-1} = \left\{p_1^{K-1}, \ldots, p_N^{K-1}\right\}$ obtained from the$(k-1)$-th iteration are utilized as weights for weighted sampling, aiming to derive a set S comprising points more likely to be inliers. In lines 4 to 6, the algorithm obtains the hypothesis model θ^k, the number of outliers O^k, and the inlier classification C^k. It then finds the current best hypothesis model θ^* and inlier classification C^*, along with the minimum number of outliers O^* that maximizes the number of inliers. In line 7, the algorithm computes the inlier probabilities $\boldsymbol{P}^k = \left\{p_1^k, \ldots, p_n^k\right\}$ for each data point in the k-th iteration based on the Bayesian network model, with the computation details available in [36]. Line 8 counts the number of outliers $\overset{\sim k}{O}$, defined as those points with inlier probabilities p_i^k below a specified threshold τ (with $\tau = 0.01$). In line 9, the stopping criterion is triggered when $\overset{\sim k}{O} \geqq O^*$, and the loop is exited. This indicates that the number of outliers in the current model θ^k is greater than or equal to the minimal outlier count O^* corresponding to the best hypothetical model θ^*, implying a significantly reduced likelihood of obtaining fewer outliers through further iterations. The loop is thus terminated, and the current optimal model hypothesis θ^* and inlier classification C^* are output.

After tracking with the LK optical flow, we use the algorithm to compute the essential matrix and solve the Perspective-n-Point (PnP) problem, aiming to mitigate the impact of erroneous tracking points on the system's state.

5.2. Frame-to-Frame Reprojection Error

In the previous frame F^{k-1}, assuming m map points $\boldsymbol{M} = \{\boldsymbol{P}_1, \ldots, \boldsymbol{P}_m\}$ were tracked, with their projection coordinates in the previous frame being $\{\rho_{1_{k-1}}, \ldots, \rho_{m_{k-1}}\}$, using the LK optical flow method, the pixel coordinates of these map points in the current frame F^k are obtained as $\{\rho_{1_k}, \ldots, \rho_{m_k}\}$. The reprojection error is defined as the Euclidean distance between the projection coordinates from the previous frame and the current projections, as illustrated by the geometric constraint in Figure 4. By formulating a nonlinear optimization problem, the ESIKF algorithm is employed to minimize the reprojection errors, thereby optimizing the system's state.

For the s-th map point ${}^G\boldsymbol{P}_s = \left[{}^G\boldsymbol{p}_s^T, \boldsymbol{c}_s^T\right]^T = \left[{}^Gp_{sx}, {}^Gp_{sy}, {}^Gp_{sz}, c_{sr}, c_{sg}, c_{sb}\right]^T, {}^G\boldsymbol{P}_s \in \boldsymbol{M}$, where ${}^G\boldsymbol{p}_s^T$ represents the three-dimensional coordinates of the map point, we project this map point into the camera coordinate system and obtain its reprojection error $r\left(\hat{\boldsymbol{x}}_k, \rho_{s_k}, {}^G\boldsymbol{p}_s\right)$ using Equation (19).

$$ {}^C\boldsymbol{p}_s = \left[{}^Cp_{sx}, {}^Cp_{sy}, {}^Cp_{sz}\right]^T = \left({}^G\hat{\boldsymbol{R}}_{I_k} \cdot {}^I\hat{\boldsymbol{R}}_{C_k}\right)^T \cdot {}^G\boldsymbol{p}_s - {}^I\hat{\boldsymbol{R}}_{C_k}^T \cdot {}^I\hat{\boldsymbol{p}}_{C_k} - \left({}^G\hat{\boldsymbol{R}}_{I_k} \cdot {}^I\hat{\boldsymbol{R}}_{C_k}\right)^T \cdot {}^G\hat{\boldsymbol{p}}_{I_k} \tag{24} $$

$$ r\left(\hat{\boldsymbol{x}}_k, \rho_{s_k}, {}^G\boldsymbol{p}_s\right) = I_k(\rho_{s_k}) - I_k\left(\zeta\left({}^C\boldsymbol{p}_s, \hat{\boldsymbol{x}}_k\right)\right) \tag{25} $$

$$ \zeta\left({}^C\boldsymbol{p}_s, \hat{\boldsymbol{x}}_k\right) = \left[\hat{f}_{x_k}\frac{{}^Cp_{sx}}{{}^Cp_{sz}} + c_{x_k}\ ,\ \hat{f}_{y_k}\frac{{}^Cp_{sy}}{{}^Cp_{sz}} + c_{y_k}\right]^T + \frac{{}^I\hat{t}_{c_k}}{\Delta t_{k-1,k}}\left(\rho_{s_k} - \rho_{s_{k-1}}\right) \tag{26} $$

where ζ represents a projection function, the latter term of Equation (20) is the time correction factor, $\Delta t_{k-1,k}$ is the time interval between the previous frame F^{k-1} and the current frame F^k, and (c_{x_k}, c_{y_k}) and $\left(\hat{f}_{x_k}\hat{f}_{y_k}\right)$ are the intrinsic parameters of the camera, specifically the principal point and focal length, respectively. During the optimization process aimed at minimizing the reprojection errors, these intrinsic parameters are treated as variables to be optimized, adjusting them iteratively to enhance the accuracy of the system's state estimation.

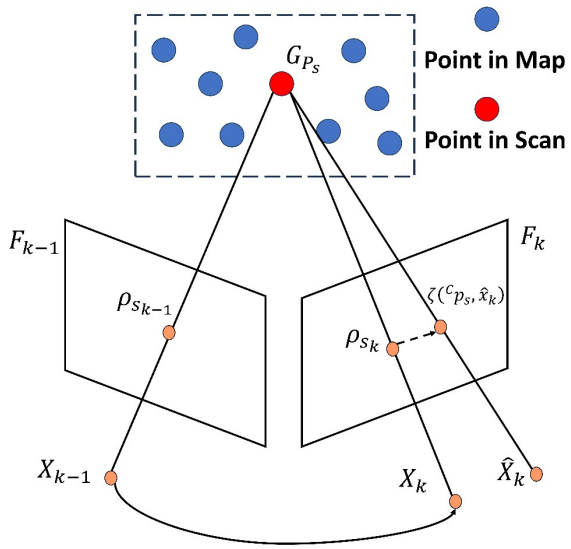


Figure 4. Illustration of reprojection error.

5.3. Frame-to-Map RGB Color Error

For the s-th map point ${}^{G}\boldsymbol{P}_s = \left[{}^{G}\boldsymbol{p}_s^T, \boldsymbol{c}_s^T\right]^T = \left[{}^{G}p_{sx}, {}^{G}p_{sy}, {}^{G}p_{sZ}, c_{sr}, c_{sg}, c_{sb}\right]^T, {}^{G}\boldsymbol{P}_s \in \boldsymbol{M}$, the RGB color error model is established through the following steps: First, the map point is projected onto the current image plane via a pinhole projection model to obtain its corresponding pixel coordinates ρ. Subsequently, a linear interpolation algorithm is applied to compute the RGB values of sampled points within the neighborhood of ρ, estimating the projected point's color descriptor γ_s (as illustrated in Figure 5). Finally, a RGB color error function $o\left(\hat{\boldsymbol{x}}_k, {}^{G}\boldsymbol{p}_s, \boldsymbol{c}_s\right)$ is constructed between the map point's color $\boldsymbol{c}_s$ and the projected point's color γ_s. This error term is integrated into a nonlinear optimization framework, where the ESIKF is employed to iteratively minimize the RGB color errors and refine the system state.

$$o\left(\hat{\boldsymbol{x}}_k, {}^{G}\boldsymbol{P}_s, \boldsymbol{c}_s\right) = \boldsymbol{c}_s - \gamma_s \tag{27}$$

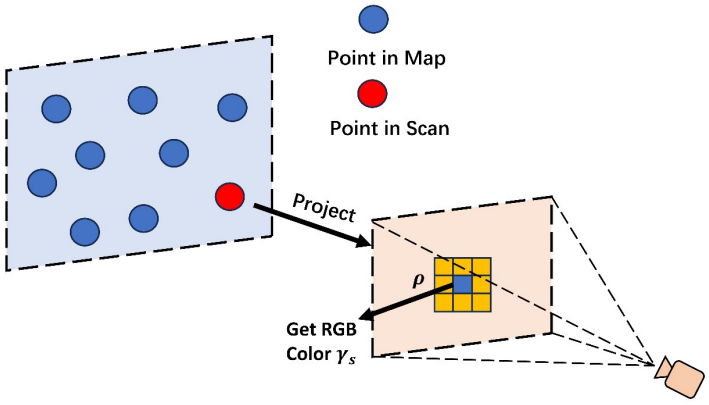


Figure 5. Illustration of obtaining RGB information, with yellow squares indicating adjacent pixels.

5.4. State Estimation Based on ESIKF

Combining the prior state estimate $\hat{\boldsymbol{x}}_k$ and its covariance $\boldsymbol{\Sigma}_{\delta\hat{\boldsymbol{x}}_k}$, obtained from IMU propagation with the frame-to-frame reprojection error and the frame-to-map RGB color error, yields a maximum a posteriori (MAP) estimate for the state vector $\boldsymbol{x}_k$.

$$\min_{\delta\check{\boldsymbol{x}}_k} \left\{ \begin{array}{c} \left\| \check{\boldsymbol{x}}_k \boxminus \hat{\boldsymbol{x}}_k + \boldsymbol{J}^k \delta\check{\boldsymbol{x}}_k \right\|^2_{\boldsymbol{\Sigma}_{\delta\hat{\boldsymbol{x}}_k}} + \sum_{s=1}^{N} \left\| r\left(\hat{\boldsymbol{x}}_k, \boldsymbol{\rho}_{s_k}, {}^{G}\boldsymbol{p}_s\right) + \boldsymbol{H}_s^r \delta\check{\boldsymbol{x}}_k \right\|^2_{\sum_{\alpha_s}} \\ + \sum_{s=1}^{N} \left\| o\left(\hat{\boldsymbol{x}}_k, {}^{G}\boldsymbol{p}_s, \boldsymbol{c}_s\right) + \boldsymbol{H}_s^o \delta\check{\boldsymbol{x}}_k \right\|^2_{\sum_{\beta_s}} \end{array} \right\} \tag{28}$$

Here, $\boldsymbol{H}_s^r$ denotes the Jacobian matrix of the reprojection error, with corresponding observation noise covariance $\sum_{\alpha_s}$. $\boldsymbol{H}_s^o$ denotes the Jacobian matrix of the RGB color error, with corresponding observation noise covariance $\sum_{\beta_s}$. The computation of these Jacobian and covariance matrices is given below.

$$H_s^r = \frac{\partial r\left(\check{\boldsymbol{x}}_k \boxplus \delta\check{\boldsymbol{x}}_k, \boldsymbol{\rho}_{s_k}, {}^{G}\boldsymbol{p}_s\right)}{\partial \delta_{\check{\boldsymbol{x}}_k}} \tag{29}$$

$$\sum_{\alpha_s} = \sum_{n_{\rho_{s_k}}} + J^r_{{}^{G}p_s} \sum_{{}^{G}p_s} J^{r}_{{}^{G}p_s}{}^{T}, \quad J^r_{{}^{G}p_s} = \frac{\partial r\left(\check{\boldsymbol{x}}_k, \rho_{s_k}, {}^{G}\boldsymbol{p}_s\right)}{\partial {}^{G}\boldsymbol{p}_s} \tag{30}$$

$$H_s^o = \frac{\partial o\left(\check{\boldsymbol{x}}_k \boxplus \delta\check{\boldsymbol{x}}_k, {}^{G}\boldsymbol{p}_s, \boldsymbol{c}_s\right)}{\partial \delta_{\check{\boldsymbol{x}}_k}} \tag{31}$$

$$\sum_{\beta_s} = \sum_{n_{c_s}} + \sum_{n_{\gamma_s}} + J^o_{{}^{G}p_s} \sum_{{}^{G}p_s} J^{o}_{{}^{G}p_s}{}^{T}, \quad J^o_{{}^{G}p_s} = \frac{\partial o\left(\check{\boldsymbol{x}}_k, {}^{G}\boldsymbol{p}_s, \boldsymbol{c}_s\right)}{\partial {}^{G}\boldsymbol{p}_s} \tag{32}$$

where $\sum_{G p_s}$ is the observation error covariance of the point, $\sum_{n_{\rho s_k}}$ is the reprojection observation error covariance, and $\sum_{n_{cs}}$ and $\sum_{n_{\gamma s}}$ are the RGB color observation error covariances for the map point and the pixel point, respectively. The Kalman gain in Equation (18) is further calculated using the following formula:

$$\boldsymbol{H} = \left[\boldsymbol{H}_1^{r\,T}, \ldots, \boldsymbol{H}_m^{r\,T}, \boldsymbol{H}_1^{o\,T}, \ldots, \boldsymbol{H}_m^{o\,T} \right]^T \tag{33}$$

$$\boldsymbol{R} = diag\left(\boldsymbol{\Sigma}_{\alpha_1}, \ldots, \boldsymbol{\Sigma}_{\alpha_m}, \boldsymbol{\Sigma}_{\beta_1}, \ldots, \boldsymbol{\Sigma}_{\beta_m} \right) \tag{34}$$

$$\boldsymbol{P} = \left(\boldsymbol{J}^k \right)^{-\mathbf{1}} \boldsymbol{\Sigma}_{\delta \hat{\boldsymbol{x}}_k} \left(\boldsymbol{J}^k \right)^{-\boldsymbol{T}} \tag{35}$$

$$\check{\boldsymbol{z}}_k = \left[r\left(\hat{\boldsymbol{x}}_k, \boldsymbol{\rho}_{1_k}, {}^G\boldsymbol{p}_1 \right), \ldots, r\left(\hat{\boldsymbol{x}}_k, \boldsymbol{\rho}_{m_k}, {}^G\boldsymbol{p}_m \right), o\left(\hat{\boldsymbol{x}}_k, {}^G\boldsymbol{p}_1, \boldsymbol{c}_1 \right), \ldots, o\left(\hat{\boldsymbol{x}}_k, {}^G\boldsymbol{p}_m, \boldsymbol{c}_, \right) \right]^T \tag{36}$$

The system subsequently updates the state vector through Equation (21). When the iterative process satisfies the convergence criteria, it performs the final state update using Equation (23). Upon convergence, the state vector simultaneously performs the following critical operations: (1) it updates global map points and tracking points, and (2) it serves as the initial state for IMU propagation in either the LIO or VIO subsystem during the next frame.

6. Map Management

(1) LiDAR Submap: In the LiDAR submap, raw point clouds are partitioned into fixed-size voxel grids for processing. When the point cloud within a voxel forms a valid planar feature, the system records the geometric parameters of the plane to support subsequent point-to-plane matching and voxel merging operations. Voxel management is implemented via a union-find data structure combined with a hash table for efficient spatial indexing. To prevent computational overload, a maximum capacity threshold of 50 point clouds per voxel is enforced. Once this storage limit is reached, the voxel will stop updating and clear the internal point cloud data, retaining only the calculated plane feature parameters.

(2) Global Map: For the global map point ${}^G\boldsymbol{P}_s = \left[{}^G p_{sx}, {}^G p_{sy}, {}^G p_{sz}, c_{sr}, c_{sg}, c_{sb} \right]^T$, the global map stores not only the position of the map point in the global coordinate system but also the RGB information of the map point. After the system state is updated using the ESIKF in the VIO system, a color fusion strategy based on Bayesian inference is employed. This strategy probabilistically integrates the color observations obtained via projection from the current image frame with the prior color stored in the map points. The weighted update mechanism in this process significantly enhances the accuracy of color estimation. Subsequently, based on the updated system state from ESIKF, the projection operation of the map point to the current image plane is re-executed, and the reprojection error and RGB color error in the current image frame are calculated. If the error exceeds a set threshold, the tracked point is discarded to ensure the accuracy of the system state and maintain the consistency of the map. This approach not only enhances the system's robustness under complex lighting conditions but also reduces the impact of errors caused by incorrect tracking points on system localization and mapping.

7. Experiments and Analysis

To validate the superiority of the proposed method, experiments were conducted on a computer configured with a 12-core Intel® Core™ i5-10400F CPU @ 2.9 GHz, 32 GB RAM, RTX 2080Ti, and Ubuntu 20.04. The experiments utilized the M2DGR [42] and

NTU-VIRAL [43] datasets, comparing the proposed method's localization performance against several state-of-the-art algorithms, including FAST-LIO2, LIO-SAM, Voxelmap++, LVI-SAM, FAST-LIVO, and R3Live. Localization accuracy was quantified using the absolute trajectory error (ATE) [44] metric, computed with the open-source evaluation tool evo. The ATE is defined as the Euclidean distance between the estimated and ground truth trajectories after optimal alignment, providing a measure of the algorithm's precision in trajectory estimation.

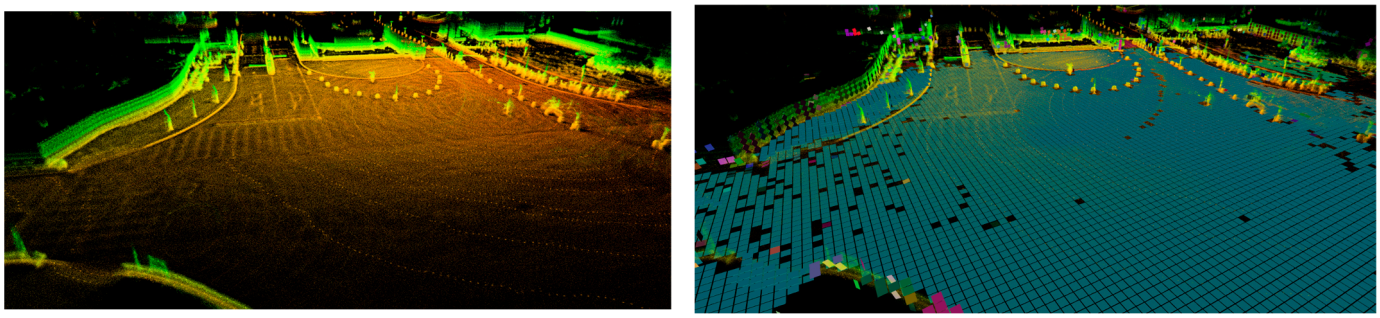
$$ATE = \sqrt{\frac{1}{N}\sum_{i=1}^{N}\left\|\boldsymbol{t}_{est}^{i} - \boldsymbol{t}_{gt}^{i}\right\|} \tag{37}$$

where $\boldsymbol{t}_{est}^{i}$ is the estimated pose and $\boldsymbol{t}_{gt}^{i}$ is the ground truth pose. Since the proposed method does not include a loop closure detection module, for the sake of experimental fairness, the loop closure detection modules of LIO-SAM and LVI-SAM were disabled in the subsequent experiments.

7.1. Experiments on the M2DGR Datasets

The M2DGR dataset was collected using a mobile robotic platform equipped with multiple sensors, including the Velodyne VLP-32C LiDAR, Realsense d435i camera, and Handsfree A9 IMU, facilitating data acquisition in complex indoor and outdoor environments. Ground truth trajectories were obtained using motion capture devices, a laser 3D tracker, and an RTK receiver. As shown in Table 1, the proposed method achieved optimal performance across most sequences, with an average absolute trajectory error (ATE) root mean square error (RMSE) of 0.478 m. Notably, due to strict data synchronization requirements, the FAST-LIVO algorithm encountered compatibility issues with the M2DGR dataset, and thus, its results are not included in Table 1.

In indoor sequences such as *door_*, *hall_*, and *room_**, the proposed method performed exceptionally well, attributed to the abundant continuous planar structures in indoor environments that provide strong geometric constraints for pose estimation. In LiDAR point cloud processing, the probabilistic plane modeling approach discretizes raw data into fixed-size voxel grids, causing large-scale continuous planes to be fragmented into multiple small coplanar voxel planes. To enhance point cloud registration efficiency, a voxel plane merging strategy was implemented to merge coplanar voxel planes (as shown in Figure 6). The fused plane features offer effective geometric constraints for pose estimation in SLAM systems and improve the matching efficiency between LiDAR point clouds and planes.



(**a**) (**b**)

Figure 6. (**a**) The point cloud map corresponding to the *gate_01* sequence. (**b**) The representation of voxel planes in the *gate_01* sequence. Each square represents a voxel plane, and voxel planes that are coplanar share the same color.

Table 1. RMSE of ATE in M2DGR datasets in meters.

Sequence	Features	Duration(s)	FAST-LIO2	LIO-SAM	Voxelmap++	LVI-SAM	R3LIVE	Our Method
door_01	Outdoor to indoor to outdoor, long-term	461	0.407	0.246	0.244	0.245	0.247	**0.187**
door_02	Outdoor to indoor	127	0.28	0.184	0.194	0.185	0.274	**0.177**
gate_01	Dark, around gate	172	0.174	0.635	0.163	0.142	0.228	**0.11**
gate_02	Dark, loop back	327	**0.32**	0.341	0.526	0.346	0.38	0.335
gate_03	Day	283	0.112	0.106	0.148	0.104	0.104	**0.083**
hall_01	Random walk	351	0.284	0.236	0.27	0.241	0.258	**0.213**
hall_02	Random walk	128	0.513	0.278	0.251	0.273	0.372	**0.204**
hall_03	Random walk	164	0.573	0.466	0.282	0.324	0.352	**0.196**
hall_04	Random walk	181	1.045	0.914	0.892	0.849	0.938	**0.725**
hall_05	Random walk	402	1.18	1.011	0.999	1.03	1.03	**0.793**
room_01	Room, bright	72	0.312	0.159	0.134	0.135	0.203	**0.079**
room_02	Room, bright	75	0.315	0.126	0.12	0.127	0.199	**0.088**
room_03	Room, bright	128	0.413	0.162	0.161	**0.152**	0.201	0.16
street_02	Day, long-term	1227	3.096	3.564	fail	3.46	**2.834**	3.943
street_03	Night, back and forth, full speed	354	0.177	0.508	fail	0.131	0.664	**0.099**
street_04	Night, around lawn, loop back	858	0.464	0.832	fail	0.924	**0.302**	0.551
street_05	Night, straight line	469	**0.299**	0.337	3.255	0.337	0.385	0.317
street_06	Night, one turn	494	0.364	0.386	fail	0.379	0.368	**0.342**
Mean		348.5	0.574	0.583	1.628	0.521	0.519	**0.478**

Note: "fail" refers to cases where the absolute trajectory error exceeds 10. Bold font indicates the best performance.

In outdoor scenarios, the algorithm's performance on the *street_** sequences datasets shows a declining trend. This is primarily due to the sparse distribution of map feature points and a lack of continuous planar features. Notably, in extended sequences such as *street_02* and *street_04*, there is a significant degradation in performance. Analysis indicates that the voxel plane merging operation leads to a cumulative error effect, which is the main contributor to the decline in system accuracy. However, in shorter-duration sequences, the proposed method demonstrates good adaptability. Comparative experiments with Voxelmap++ show that the VIO subsystem effectively corrects cumulative errors from the LIO subsystem, enhancing both positioning accuracy and robustness of the overall system.

Figure 7 illustrates the comparison between the estimated trajectory of the proposed method and the ground truth trajectory in the gate_01 sequence. As demonstrated in Figures 8 and 9, our method exhibits significant advantages in trajectory fitting accuracy and localization error control compared to other algorithms, with the estimated trajectory closely aligning with the ground truth. However, Figure 8b reveals transient abrupt deviations in the algorithm's trajectory during intense rotational motion, followed by rapid recovery. This occurs because excessively large rotational displacements hinder point clouds from identifying sufficient voxel planes for registration within a short timeframe, occasionally leading to erroneous merging of non-coplanar planes. Concurrently, motion blur induced by high-speed movement degrades the VIO subsystem's localization performance. Nevertheless, such fluctuations remain temporary. Once motion stabilizes, the LIO subsystem promptly corrects the pose by identifying voxel planes coplanar with the current LiDAR point cloud in the radar submap. Overall, this phenomenon indicates that the method maintains robust performance under aggressive motion, though further optimization is warranted to address abrupt kinematic transitions.

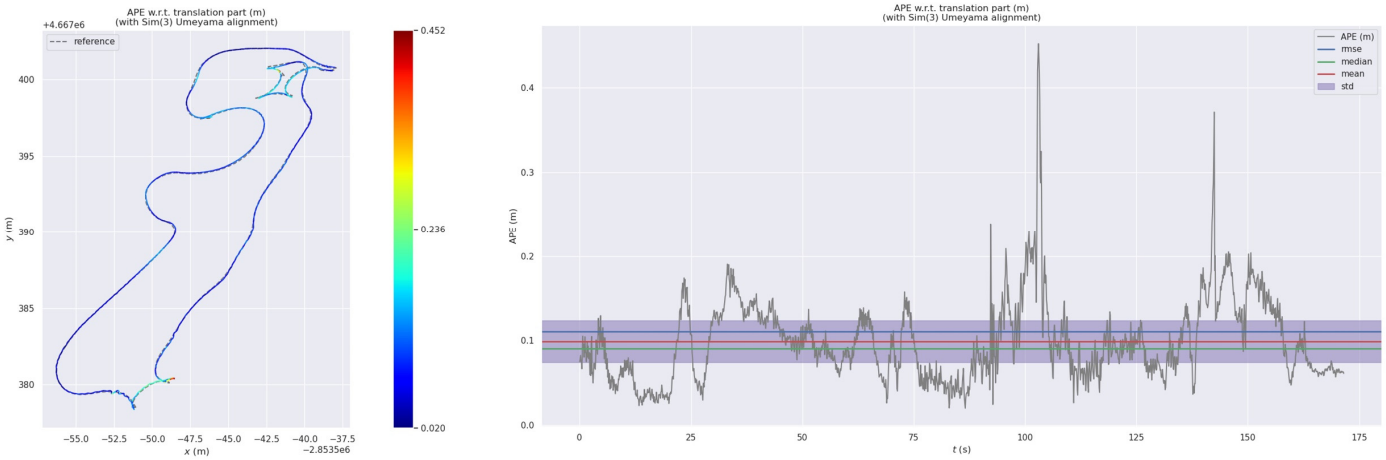


Figure 7. Trajectory estimation and its difference from the ground truth in the *gate_01* sequence for the proposed method.

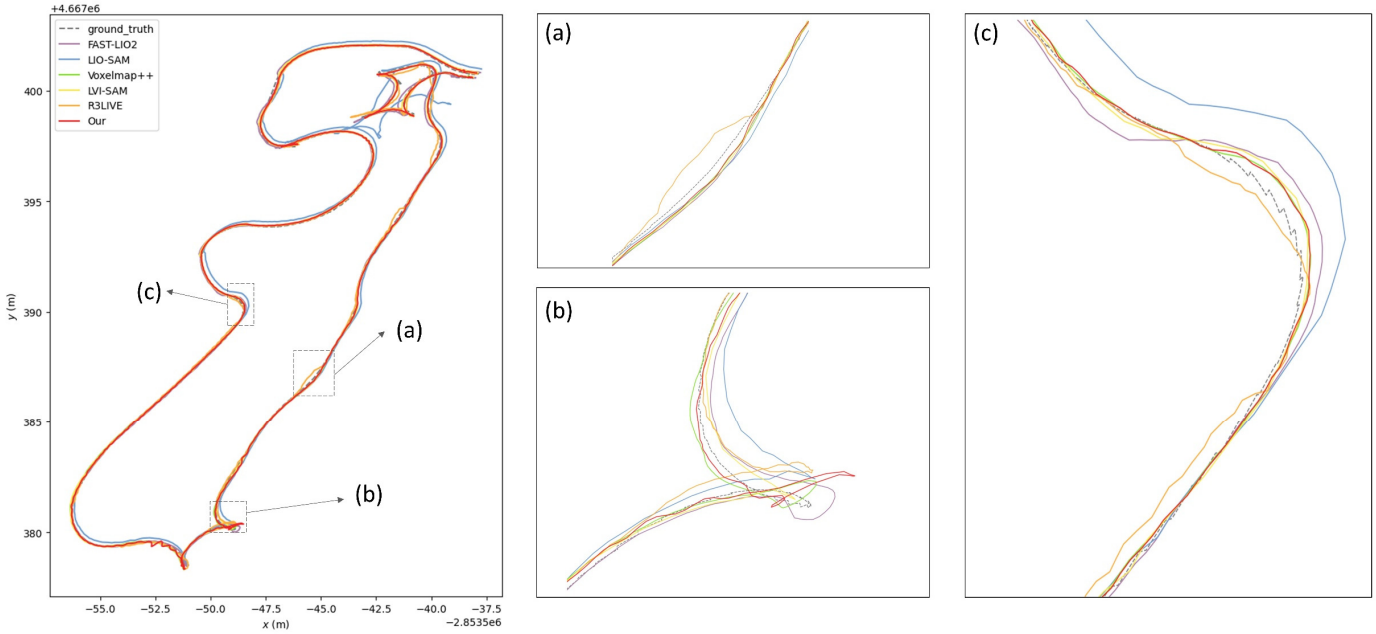


Figure 8. Trajectory comparison in the *gate_01* sequence. (**a**–**c**) show magnified views of the trajectory.

To further characterize the performance of the proposed method, we conducted detailed comparisons of absolute trajectory error (ATE) variations across multiple algorithms on low-light scene sequences gate_01 and street_03, with quantitative results illustrated in Figures 10 and 11. Comparisons with existing mainstream algorithms demonstrate that the proposed method maintains superior performance in most scenarios. It should be noted that the Voxelmap++ algorithm failed to run properly on the *street_03* sequence, so its data are not included in Figure 11. Notably, Figure 10 shows that our method experiences a sudden increase in trajectory error in certain sections of the *gate_01* sequence, and an analysis of the trajectory visualization in Figure 8b indicates that this anomaly is due to violent rotational transformations present in the sequence. In contrast, the motion pattern in the *street_03* sequence is relatively stable, and as shown in Figure 11, the proposed method maintains a consistently low error level throughout this sequence.

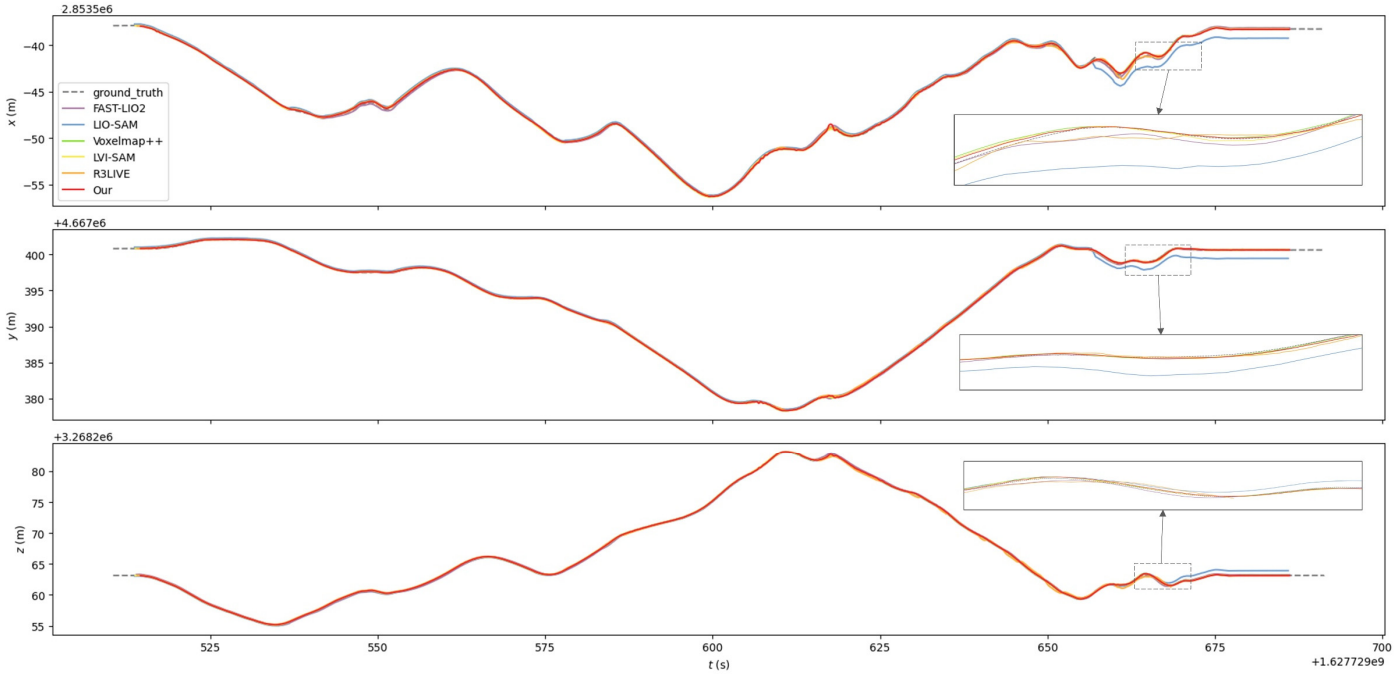


Figure 9. Localization error in the *gate_01* sequence.

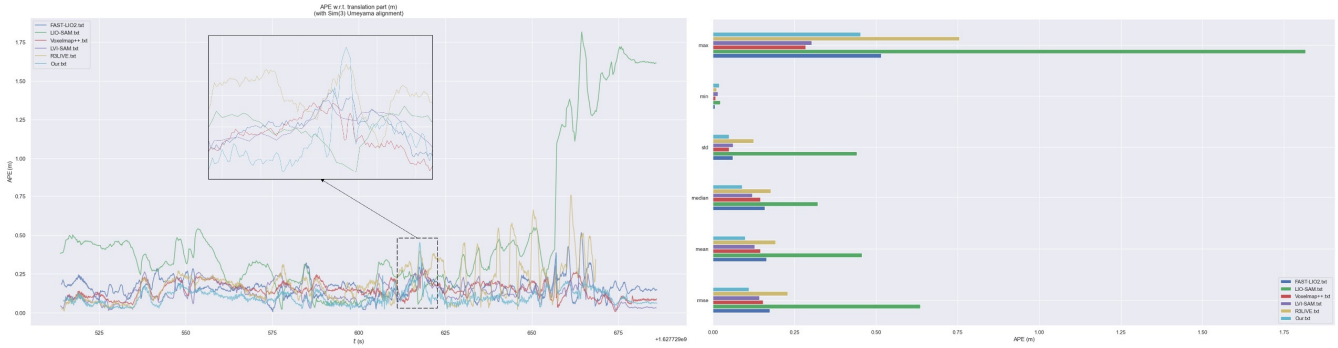


Figure 10. Variation in absolute trajectory error for different algorithms in *gate_01*.

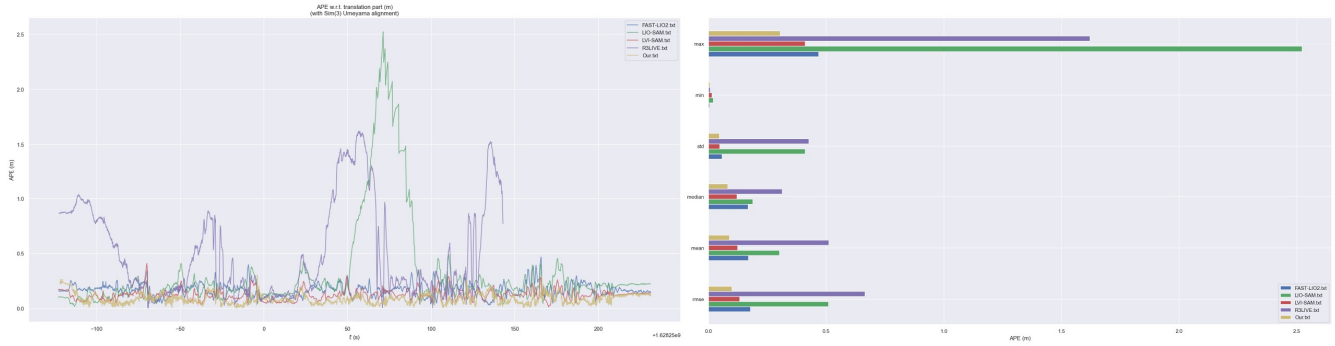


Figure 11. Variation in absolute trajectory error for different algorithms in the *street_03* sequence.

7.2. Experiments on the NTU-VIRAL Datasets

The NTU-VIRAL dataset is a comprehensive collection designed for research in autonomous aerial systems. It comprises data from multiple sensors, including two 3D LiDARs, two synchronized global-shutter cameras, several inertial measurement units (IMUs), and multiple ultra-wideband (UWB) ranging units, all mounted on a DJI Matrice 600 Hexacopter. Data were collected across various locations within Nanyang Technological University (NTU), Singapore.

As shown in Table 2, the method proposed in this study achieves optimal overall performance, with an average root mean square error (RMSE) of 0.185 m for absolute trajectory error (ATE). The experimental results indicate superior performance in the low-light indoor sequences like '*nya_**', attributed to the abundant continuous planar structures in confined spaces that provide strong geometric constraints for pose estimation. However, in outdoor sequences such as '*eee_**' and '*sbs_**', the performance is comparatively poor. This is primarily due to the scarcity of continuous planar features in open environments, which limits effective constraints for pose estimation. Additionally, the NTU-VIRAL dataset captures grayscale images of the surrounding environment, which limits the full exploitation of the frame-to-map RGB color alignment module in VIO. Despite these limitations, the proposed method still achieves a significant improvement in localization accuracy compared to R3LIVE, which also utilizes an RGB color alignment module.

Table 2. RMSE of ATE in NTU-VIRAL datasets in meters.

Sequence	Features	Duration(S)	FAST-LIO2	LIO-SAM	Voxelmap++	FAST-LIVO	LVI-SAM	R3LIVE	Our Method
eee_01	Outdoor, bright	398.7	0.222	0.193	0.198	0.27	0.179	0.219	**0.147**
eee_02	Outdoor, bright	321.1	0.158	**0.117**	0.216	0.162	0.216	0.736	0.175
eee_03	Outdoor, bright	181.4	0.2208	**0.19**	0.214	0.278	0.246	0.207	0.227
nya_01	Outdoor, square	396.3	0.245	0.205	0.433	0.276	0.204	0.302	**0.153**
nya_02	Outdoor, square	428.7	0.231	0.181	0.176	0.237	0.182	0.222	**0.131**
nya_03	Outdoor, square	411.2	0.254	0.263	0.988	0.257	**0.153**	0.1677	0.227
sbs_01	Indoor, low lighting	354.2	0.265	0.312	0.207	0.531	**0.206**	0.662	0.239
sbs_02	Indoor, low lighting	373.3	0.253	**0.203**	0.397	0.326	0.204	2.063	0.237
sbs_03	Indoor, low lighting	389.3	0.249	0.277	0.167	0.223	0.268	0.152	**0.126**
Mean		361.58	0.2331	0.212	0.333	0.284	0.206	0.5256	**0.185**

Note: Bold font indicates the best performance.

7.3. Visualization for Maps

The M2DGR system employs the mechanically rotating Velodyne VLP-32C LiDAR, which uses 32 laser beams to achieve a 360° horizontal scan. As shown in Figure 6a, maps built using this sensor alone cannot intuitively capture the rich detail of the surrounding environment, nor can they fully demonstrate the performance of our mapping algorithm. Moreover, the NTU-VIRAL dataset lacks RGB information and thus cannot adequately reflect the quality of a global RGB map. To obtain both higher-density point cloud detail and full-color mapping, this chapter instead uses the hku_campus_seq_02 sequence from the R3LIVE dataset. This sequence is collected with a Livox Avia LiDAR—which offers greater point cloud density to capture finer features—and an RGB camera, fully satisfying the dual requirements of detailed geometry and color reconstruction. The LIO subsystem's mapping results on this sequence are presented in Figure 12.

Additionally, this paper conducts a comparative analysis between the proposed algorithm and R3LIVE regarding RGB global map reconstruction on the hku_campus_seq_02 dataset (see Figure 13a,b). The results demonstrate that the RGB maps generated by our algorithm exhibit slightly reduced detail clarity but overall superior performance compared to R3LIVE.

To further evaluate the generalizability of our algorithm, we performed comparative experiments on the NTU-VIRAL dataset's eee_01 sequence, which utilizes only grayscale images. Experimental findings reveal comparable mapping performance between the two algorithms, with no significant performance disparity observed (see Figure 13c,d). Comprehensive experimental results confirm that the proposed algorithm effectively constructs globally consistent mapping frameworks while maintaining real-time operational capabilities and localization precision.

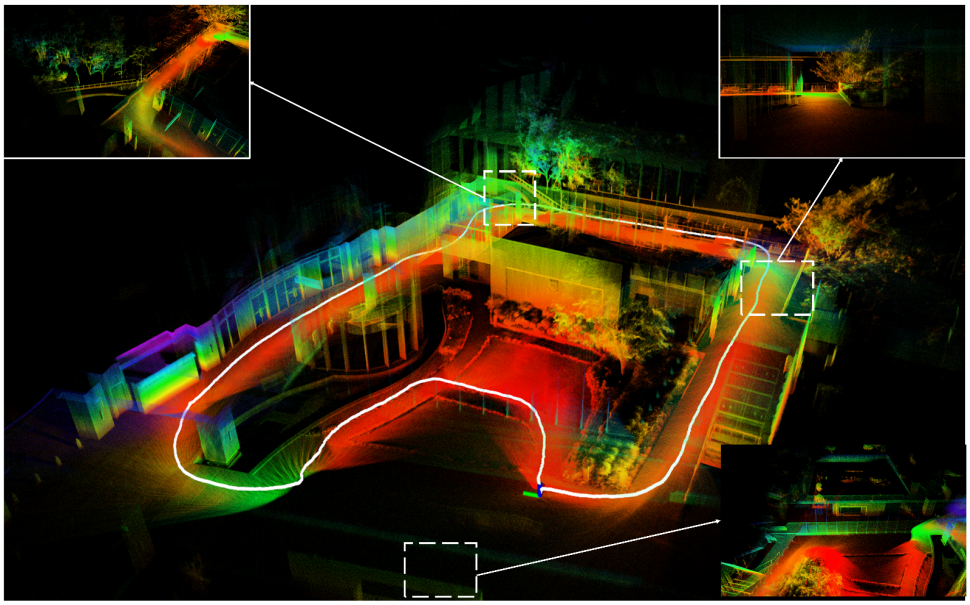
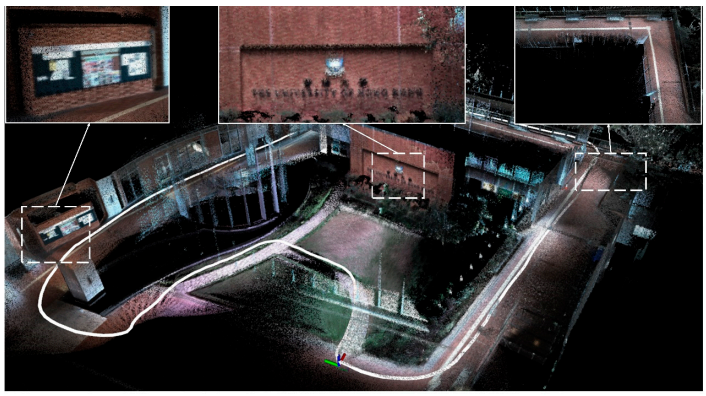
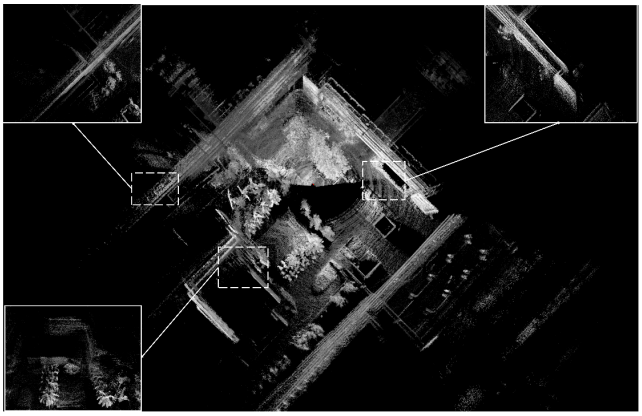


Figure 12. Mapping result of the proposed algorithm's LIO subsystem on the hku_campus_seq_02 sequence.

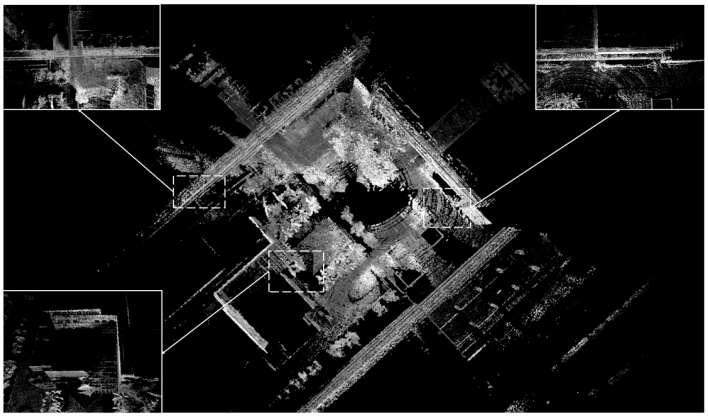
(**a**) Our method



(**b**) R3LIVE



(**c**) Our method



(**d**) R3LIVE

Figure 13. Comparison between the proposed method and R3LIVE in RGB global map construction.

7.4. Running Time Analysis

This study evaluates the real-time performance of FAST-LIVO, R3LIVE, and the proposed method across all sequences in the NTU-VIRAL dataset. Experimental results are presented in Table 3. The results demonstrate that the proposed method achieves superior real-time performance with an average computational time consumption of 39.19 ms.

Benefiting from the 3DoF incremental voxel mapping with sub-plane merging and $O(1)$ time complexity enabled by hash-table-based queries, the LIO subsystem of our method achieves optimal processing efficiency, requiring only 20.39 ms per frame on average.

Table 3. Average time consumption on the NTU-VIRAL datasets.

Sequence	FAST-LIVO		R3LIVE		Our Method	
	LIO	VIO	LIO	VIO	LIO	VIO
eee_01	24.41	**8.74**	26.23	29.73	**20.4**	15.26
eee_02	25.12	**9.14**	27.34	30.01	**20.14**	15.28
eee_03	24.22	**8.63**	25.51	29.68	**18.33**	13.95
nya_01	23.85	**8.92**	28.68	31.35	**19.85**	14.7
nya_02	24.08	**9.33**	29.31	31.94	**21.08**	15.22
nya_03	24.57	**8.96**	29.22	31.27	**20.21**	15.53
sbs_01	23.69	**9.98**	28.02	29.31	**18.76**	16.74
sbs_02	23.39	**9.34**	28.64	29.57	**18.39**	15.59
sbs_03	23.21	**9.47**	28.92	29.42	**19.03**	16.36
Mean	24.06	**9.17**	27.99	30.25	**20.39**	15.8
Total	**33.23**		58.24		36.19	

Note: Bold font indicates the best performance.

In contrast, both FAST-LIVO and R3LIVE employ incremental KD-Tree structures for map management in their LIO subsystems. This approach incurs an $O(NlogN)$ time complexity for voxel map updates, where N denotes the number of point clouds, primarily due to nearest-neighbor searches and dynamic balancing operations of tree structures. The proposed method eliminates complex neighbor search operations through union-find structures and hash tables for point cloud map management. While maintaining localization accuracy, this innovation reduces time complexity to $O(N)$, significantly enhancing computational efficiency in the LIO subsystem.

For image frames of size 752 $\times$ 480 in the dataset, the VIO subsystem of the proposed method demonstrates improved computational speed compared to R3LIVE. This enhancement stems from two key optimizations: (1) replacement of conventional RANSAC with a computationally efficient dynamic-Bayesian-network-based RANSAC method, and (2) strict inclusion of LiDAR points from LIO-identified planar voxels into the global map. However, the VIO subsystem of the proposed method exhibits slower computation than FAST-LIVO's VIO module, as our method requires additional computations for frame-to-map RGB error minimization and global map maintenance. Consistent with R3LIVE, our method updates the global RGB map during each iteration with $O(m)$ time complexity (where m represents the number of map points), resulting in relatively higher computational overhead.

8. Conclusions

This paper proposes a multi-sensor fusion framework that efficiently couples LiDAR, IMU, and camera data through mergeable probabilistic voxel mapping. In the LIO subsystem, after removing motion distortion from LiDAR point clouds using IMU pre-integration, the uncertainty of map points caused by noise in the LiDAR data is modeled probabilistically using voxel maps to construct voxel planes. Furthermore, the 6DoF plane parameters are optimized to 3DoF plane parameters to enhance computational efficiency. Additionally, the union-find data structure and hash table are employed to merge sub-planes that share coplanar relationships within the same parent plane. Experimental results show that this approach effectively accelerates the point-to-plane matching speed. In the VIO subsystem, the state estimation from the LIO subsystem is used as an initial guess, with IMU

pre-integration providing a rough estimate of the system state. The global map is projected onto image frames to directly acquire tracking points. The random sample consensus algorithm based on a dynamic Bayesian network is then employed to eliminate outlier tracking points. The system state is further refined by minimizing both frame-to-frame reprojection errors and frame-to-map RGB color errors, while simultaneously estimating and optimizing the camera intrinsic parameters online. Experiments on the public datasets M2DGR and NTU-VIRAL demonstrate that the proposed method achieves effective results in both localization accuracy and time consumption. However, the accumulation of errors due to voxel plane merging, which was not effectively corrected, leads to suboptimal performance on long sequences.

In future work, improvements should be pursued in the following aspects. First, the planar merging method employed in this study neglects the heterogeneity between individual voxel planes, potentially leading to erroneous merging of non-coplanar surfaces. Subsequent research should explore superior plane fusion strategies to mitigate cumulative errors. Specifically, an adaptive weighting mechanism could be introduced to assign varying confidence levels based on planar observation uncertainties and geometric characteristics, thereby enhancing merging decision accuracy.

Second, the proposed method exhibits significant robustness degradation under aggressive rotational motions. This limitation could be addressed by analyzing voxel variation patterns to optimize algorithmic adaptability. Implementing an inertial measurement unit (IMU)-based motion prediction model could alleviate this issue through dynamic adjustment of point cloud registration parameters using rotational rate information, thereby enhancing system stability in complex motion scenarios.

Finally, while the current model accounts for LiDAR measurement noise and visual errors induced by camera intrinsic parameter variations, future efforts should integrate direct measurement noise factors such as camera exposure duration. This necessitates developing a more comprehensive sensor noise model that incorporates exposure inconsistency, illumination variations, and motion blur into an error compensation framework, ultimately improving measurement precision and consistency in challenging environments.

Author Contributions: Conceptualization, B.W., N.B., H.X., X.Z. and H.L.; data curation, B.W.; formal analysis, B.W., N.B., H.X. and H.L.; funding acquisition, H.X. and X.Z.; investigation, B.W. and N.B.; methodology, B.W., N.B. and H.L.; project administration, H.X. and X.Z.; resources, B.W.; software, B.W. and N.B.; validation, B.W.; writing—original draft, B.W. and N.B.; writing—review and editing, H.X. and X.Z. All authors have read and agreed to the published version of the manuscript.

Funding: This work was supported by the National Natural Science Foundation of China (62376252), the Key Project of the Natural Science Foundation of Zhejiang Province (LZ22F030003), Zhejiang Province Leading Geese Plan (2025C02025, 2025C01056), and Zhejiang Province Province–Land Synergy Program (2025SDXT004-3).

Data Availability Statement: Data are contained within the article.

Conflicts of Interest: Author Xinzhong Zhu and Hongbo Li were employed by the company Beijing Geekplus Technology Co., Ltd. The remaining authors declare that the research was conducted in the absence of any commercial or financial relationships that could be construed as a potential conflict of interest.

References

1. Levinson, J.; Askeland, J.; Becker, J.; Dolson, J.; Held, D.; Kammel, S.; Kolter, J.Z.; Langer, D.; Pink, O.; Pratt, V. Towards fully autonomous driving: Systems and algorithms. In Proceedings of the 2011 IEEE Intelligent Vehicles Symposium (IV), Baden, Germany, 5–9 June 2011; pp. 163–168.
2. Singandhupe, A.; La, H.M. A review of slam techniques and security in autonomous driving. In Proceedings of the 2019 Third IEEE International Conference on Robotic Computing (IRC), Naples, Italy, 25–27 February 2019; pp. 602–607.
3. Beinschob, P.; Reinke, C. Graph SLAM based mapping for AGV localization in large-scale warehouses. In Proceedings of the 2015 IEEE International Conference on Intelligent Computer Communication and Processing (ICCP), Cluj-Napoca, Romania, 3–5 September 2015; pp. 245–248.
4. Gao, F.; Wu, W.; Gao, W.; Shen, S. Flying on point clouds: Online trajectory generation and autonomous navigation for quadrotors in cluttered environments. *J. Field Robot.* **2019**, *36*, 710–733. [CrossRef]
5. Davison, A.J.; Reid, I.D.; Molton, N.D.; Stasse, O. MonoSLAM: Real-time single camera SLAM. *EEE Trans. Pattern Anal. Mach. Intell.* **2007**, *29*, 1052–1067. [CrossRef] [PubMed]
6. Klein, G.; Murray, D. Parallel tracking and mapping for small AR workspaces. In Proceedings of the 2007 6th IEEE and ACM International Symposium on Mixed and Augmented Reality, Nara, Japan, 13–16 November 2007; pp. 225–234.
7. Mur-Artal, R.; Montiel, J.M.M.; Tardos, J.D. ORB-SLAM: A versatile and accurate monocular SLAM system. *EEE Trans. Robot.* **2015**, *31*, 1147–1163. [CrossRef]
8. Mur-Artal, R.; Tardós, J.D. Orb-slam2: An open-source slam system for monocular, stereo, and rgb-d cameras. *IEEE Trans. Robot.* **2017**, *33*, 1255–1262. [CrossRef]
9. Campos, C.; Elvira, R.; Rodríguez, J.J.G.; Montiel, J.M.; Tardós, J.D. Orb-slam3: An accurate open-source library for visual, visual–inertial, and multimap slam. *EEE Trans. Robot.* **2021**, *37*, 1874–1890. [CrossRef]
10. Mourikis, A.I.; Roumeliotis, S.I. A multi-state constraint Kalman filter for vision-aided inertial navigation. In Proceedings of the 2007 IEEE International Conference on Robotics and Automation, Roma, Italy, 10–14 April 2007; pp. 3565–3572.
11. Qin, T.; Li, P.; Shen, S. Vins-mono: A robust and versatile monocular visual-inertial state estimator. *IEEE Trans. Robot.* **2018**, *34*, 1004–1020. [CrossRef]
12. Newcombe, R.A.; Lovegrove, S.J.; Davison, A.J. DTAM: Dense tracking and mapping in real-time. In Proceedings of the 2011 International Conference on Computer Vision, Barcelona, Spain, 6–13 November 2011; pp. 2320–2327.
13. Forster, C.; Pizzoli, M.; Scaramuzza, D. SVO: Fast semi-direct monocular visual odometry. In Proceedings of the 2014 IEEE International Conference on Robotics and Automation (ICRA), Hong Kong, China, 31 May–7 June 2014; pp. 15–22.
14. Engel, J.; Schöps, T.; Cremers, D. LSD-SLAM: Large-scale direct monocular SLAM. In Proceedings of the European Conference on Computer Vision, Zurich, Switzerland, 6–12 September 2014; pp. 834–849.
15. Zhang, J.; Singh, S. LOAM: Lidar odometry and mapping in real-time. In Proceedings of the Robotics: Science and systems, Berkeley, CA, USA, 12–16 July 2014; pp. 1–9.
16. Shan, T.; Englot, B.; Meyers, D.; Wang, W.; Ratti, C.; Rus, D. Lio-sam: Tightly-coupled lidar inertial odometry via smoothing and mapping. In Proceedings of the 2020 IEEE/RSJ international conference on intelligent robots and systems (IROS), Las Vegas, NA, USA, 25–29 October 2020; pp. 5135–5142.
17. Xu, W.; Zhang, F. Fast-lio: A fast, robust lidar-inertial odometry package by tightly-coupled iterated kalman filter. *IEEE Robot. Autom. Lett.* **2021**, *6*, 3317–3324. [CrossRef]
18. Xu, W.; Cai, Y.; He, D.; Lin, J.; Zhang, F. Fast-lio2: Fast direct lidar-inertial odometry. *IEEE Trans. Robot.* **2022**, *38*, 2053–2073. [CrossRef]
19. Bai, C.; Xiao, T.; Chen, Y.; Wang, H.; Zhang, F.; Gao, X. Faster-LIO: Lightweight tightly coupled LiDAR-inertial odometry using parallel sparse incremental voxels. *IEEE Robot. Autom. Lett.* **2022**, *7*, 4861–4868. [CrossRef]
20. Yuan, C.; Xu, W.; Liu, X.; Hong, X.; Zhang, F. Efficient and probabilistic adaptive voxel mapping for accurate online lidar odometry. *IEEE Robot. Autom. Lett.* **2022**, *7*, 8518–8525. [CrossRef]
21. Wu, C.; You, Y.; Yuan, Y.; Kong, X.; Zhang, Y.; Li, Q.; Zhao, K. VoxelMap++: Mergeable Voxel Mapping Method for Online LiDAR (-Inertial) Odometry. *IEEE Robot. Autom. Lett.* **2023**, *9*, 427–434. [CrossRef]
22. Liu, Z.; Li, H.; Yuan, C.; Liu, X.; Lin, J.; Li, R.; Zheng, C.; Zhou, B.; Liu, W.; Zhang, F. Voxel-SLAM: A Complete, Accurate, and Versatile LiDAR-Inertial SLAM System. *arXiv* **2024**, arXiv:2410.08935.
23. Cheng, J.; Zhang, L.; Chen, Q.; Hu, X.; Cai, J. A review of visual SLAM methods for autonomous driving vehicles. *Eng. Appl. Artif. Intell.* **2022**, *114*, 104992. [CrossRef]
24. Park, S.; Schöps, T.; Pollefeys, M. Illumination change robustness in direct visual slam. In Proceedings of the 2017 IEEE International Conference on Robotics and Automation (ICRA), Singapore, 29 May–3 June 2017; pp. 4523–4530.
25. Chou, C.-C.; Chou, C.-F. Efficient and accurate tightly-coupled visual-lidar slam. *IEEE Trans. Intell. Transp. Syst.* **2021**, *23*, 14509–14523. [CrossRef]

26. Zuo, X.; Geneva, P.; Lee, W.; Liu, Y.; Huang, G. Lic-fusion: Lidar-inertial-camera odometry. In Proceedings of the 2019 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS), Macau, China, 3–8 November 2019; pp. 5848–5854.
27. Zuo, X.; Yang, Y.; Geneva, P.; Lv, J.; Liu, Y.; Huang, G.; Pollefeys, M. Lic-fusion 2.0: Lidar-inertial-camera odometry with sliding-window plane-feature tracking. In Proceedings of the 2020 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS), Las Vegas, NA, USA, 25–29 October 2020; pp. 5112–5119.
28. Shan, T.; Englot, B.; Ratti, C.; Rus, D. Lvi-sam: Tightly-coupled lidar-visual-inertial odometry via smoothing and mapping. In Proceedings of the 2021 IEEE International Conference on Robotics and Automation (ICRA), Xi'an, China, 30 May–5 June 2021; pp. 5692–5698.
29. Zheng, C.; Zhu, Q.; Xu, W.; Liu, X.; Guo, Q.; Zhang, F. Fast-livo: Fast and tightly-coupled sparse-direct lidar-inertial-visual odometry. In Proceedings of the 2022 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS), Kyoto, Japan, 23–27 October 2022; pp. 4003–4009.
30. Zheng, C.; Xu, W.; Zou, Z.; Hua, T.; Yuan, C.; He, D.; Zhou, B.; Liu, Z.; Lin, J.; Zhu, F. Fast-livo2: Fast, direct lidar-inertial-visual odometry. *IEEE Trans. Robot.* **2024**, *41*, 326–346. [CrossRef]
31. Lin, J.; Zheng, C.; Xu, W.; Zhang, F. R^2 LIVE: A Robust, Real-Time, LiDAR-Inertial-Visual Tightly-Coupled State Estimator and Mapping. *EEE Robot. Autom. Lett.* **2021**, *6*, 7469–7476. [CrossRef]
32. Lin, J.; Zhang, F. R 3 LIVE: A Robust, Real-time, RGB-colored, LiDAR-Inertial-Visual tightly-coupled state Estimation and mapping package. In Proceedings of the 2022 International Conference on Robotics and Automation (ICRA), Philadelphia, PA, USA, 23–27 May 2022; pp. 10672–10678.
33. Lin, J.; Zhang, F. R^3 LIVE++: A Robust, Real-time, Radiance Reconstruction Package with a Tightly-coupled LiDAR-Inertial-Visual State Estimator. *IEEE Trans. Pattern Anal. Mach. Intell.* **2024**, *46*, 11168–11185. [CrossRef]
34. Zhang, H.; Du, L.; Bao, S.; Yuan, J.; Ma, S. LVIO-Fusion: Tightly-Coupled LiDAR-Visual-Inertial Odometry and Mapping in Degenerate Environments. *IEEE Robot. Autom. Lett.* **2024**, *9*, 3783–3790. [CrossRef]
35. Sharp, G.C.; Lee, S.W.; Wehe, D. ICP registration using invariant features. *IEEE Trans. Pattern Anal. Mach. Intell.* **2002**, *24*, 90–102. [CrossRef]
36. Piedade, V.; Miraldo, P. BANSAC: A Dynamic BAyesian Network for Adaptive SAmple Consensus. In Proceedings of the IEEE/CVF International Conference on Computer Vision, Paris, France, 4–6 October 2023; pp. 3738–3747.
37. Engel, J.; Koltun, V.; Cremers, D. Direct sparse odometry. *IEEE Trans. Pattern Anal. Mach. Intell.* **2017**, *40*, 611–625. [CrossRef]
38. De Berg, M. *Computational Geometry: Algorithms and Applications*; Springer Science & Business Media: Berlin/Heidelberg, Germany, 2000.
39. Cai, Y.; Xu, W.; Zhang, F. ikd-tree: An incremental kd tree for robotic applications. *arXiv* **2021**, arXiv:2102.10808.
40. Yuan, C.; Liu, X.; Hong, X.; Zhang, F. Pixel-level extrinsic self calibration of high resolution lidar and camera in targetless environments. *EEE Robot. Autom. Lett.* **2021**, *6*, 7517–7524. [CrossRef]
41. Lee, W.; Yang, Y.; Huang, G. Efficient multi-sensor aided inertial navigation with online calibration. In Proceedings of the 2021 IEEE International Conference on Robotics and Automation (ICRA), Xi'an, China, 30 May–5 June 2021; pp. 5706–5712.
42. Yin, J.; Li, A.; Li, T.; Yu, W.; Zou, D. M2dgr: A multi-sensor and multi-scenario slam dataset for ground robots. *IEEE Robot. Autom. Lett.* **2021**, *7*, 2266–2273. [CrossRef]
43. Nguyen, T.-M.; Yuan, S.; Cao, M.; Lyu, Y.; Nguyen, T.H.; Xie, L. Ntu viral: A visual-inertial-ranging-lidar dataset, from an aerial vehicle viewpoint. *Int. J. Robot. Res.* **2022**, *41*, 270–280. [CrossRef]
44. Zhang, Z.; Scaramuzza, D. A tutorial on quantitative trajectory evaluation for visual (-inertial) odometry. In Proceedings of the 2018 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS), Madrid, Spain, 1–5 October 2018; pp. 7244–7251.