electronics

MDPI

Article

PLM-SLAM: Enhanced Visual SLAM for Mobile Robots in Indoor Dynamic Scenes Leveraging Point-Line Features and Manhattan World Model

Jiale Liu [1] and Jingwen Luo [1,2,*]

[1] School of Information Science and Technology, Yunnan Normal University, Kunming 650500, China; 2223100027@ynnu.edu.cn
[2] Engineering Research Center of Computer Vision and Intelligent Control Technology, Department of Education of Yunnan Province, Kunming 650500, China
* Correspondence: by1503117@buaa.edu.cn

Abstract: This paper proposes an enhanced visual simultaneous localization and mapping (vSLAM) algorithm tailored for mobile robots operating in indoor dynamic scenes. By incorporating point-line features and leveraging the Manhattan world model, the proposed PLM-SLAM framework significantly improves localization accuracy and map consistency. This algorithm optimizes the line features detected by the Line Segment Detector (LSD) through merging and pruning strategies, ensuring real-time performance. Subsequently, dynamic point-line features are rejected based on Lucas–Kanade (LK) optical flow, geometric constraints, and depth information, minimizing the impact of dynamic objects. The Manhattan world model is then utilized to reduce rotational estimation errors and optimize pose estimation. High-precision line feature matching and loop closure detection mechanisms further enhance the robustness and accuracy of the system. Experimental results demonstrate the superior performance of PLM-SLAM, particularly in high-dynamic indoor environments, outperforming existing state-of-the-art methods.

Keywords: indoor dynamic scenes; mobile robot; visual SLAM; point-line features; Manhattan worlds

Citation: Liu, J.; Luo, J. PLM-SLAM: Enhanced Visual SLAM for Mobile Robots in Indoor Dynamic Scenes Leveraging Point-Line Features and Manhattan World Model. *Electronics* **2024**, *13*, 4592. https://doi.org/10.3390/electronics13234592

Academic Editor: Dah-Jye Lee

Received: 12 October 2024
Revised: 16 November 2024
Accepted: 20 November 2024
Published: 21 November 2024

1. Introduction

In recent years, vSLAM has become a research hotspot in the fields of autonomous driving and intelligent robotics. In many current SLAM algorithms for indoor scenes, most rely on corner points as feature information for environmental 3D modeling and pose tracking [1–3]. Although point features perform well in some scenes, in indoor environments with complex geometries and texture deficiencies, it is often difficult for point features to adequately describe the geometric information of the scene. In addition, randomly appearing dynamic objects in the scene can cause serious interference with feature matching, leading to a sharp decrease in the overall performance of the system. In order to improve the stability of SLAM systems in complex and dynamic environments, some researchers have attempted to improve the stability of SLAM systems by introducing line features and dynamic feature rejection techniques.

On the one hand, line features, as a powerful complement to point features, can provide richer geometric structure information, especially in building or indoor environments, which can effectively enhance the SLAM system's ability to understand and model the environment. Thus, the introduction of line features not merely improves the robustness of feature matching, but also exhibits better performance under the influence of noise and illumination variations. In some studies, by combining point features and line features [4–6], SLAM systems can achieve more accurate localization and map construction in diverse environments. On the other hand, many indoor scenes often have different moving objects,

and the dual motion of the robot and the moving target in the field-of-view (FOV) makes the SLAM system face many technical difficulties in the process of dynamic culling. If the dynamic features extracted from moving objects are directly used for robot pose estimation, it will cause significant error accumulation during the localization process, resulting in pose drift, and the constructed map will also be inconsistent with the actual environment. Therefore, it has become a cutting-edge research hotspot for how to improve the robustness and accuracy of vSLAM in dynamic environments, and many researchers have also developed different solutions by using different methods [7–11], but all of them are proposed for specific application scenes or under some assumptions. For the application requirements of indoor dynamic scenes, considering that there are many planes (such as walls, floors, and ceilings) in indoor structured scenes, the Manhattan World model assumes that these planes are vertical or horizontal. Utilizing the additional constraints of this structural regularity of the Manhattan world allows us to further improve the pose estimation accuracy and map construction performance by restricting the rotational degrees of freedom for pose estimation. Thus, this paper develops a vSLAM algorithm for mobile robots based on point-line features and the Manhattan world model, which can effectively improve the performance of mobile robots' localization and map building in indoor dynamic scenes. The main contributions of this paper are as follows:

- A merging and optimization strategy for LSD line features is proposed, and the method of separating the point-line features is used to calculate the LSD line feature descriptors, which improves the accuracy of feature matching while ensuring real-time performance.
- The accuracy of localization in indoor environments is improved by weighting the reprojection error of point-line features and optimizing the robot's pose in conjunction with the Manhattan world model.
- A dynamic point-line feature rejection method combining LK optical flow, geometric constraints, and image depth is constructed to effectively avoid the influence of dynamic objects on the system.
- On the basis of obtaining closed-loop candidate keyframes using the Bag of Words (BOW) model, geometric consistency verification is performed by calculating the similarity of line features to improve the accuracy of loop closure detection.

The rest of the paper is organized as follows. The related works about point-line features based on vSLAM and dynamic vSLAM are briefly presented in Section 2. The pipeline of the proposed framework and method are introduced in Section 3, and its implementation scheme is detailed. The simulation studies and experimental results are presented in Section 4, while Section 5 concludes the paper and provides future works.

2. Related Work

The performance of vSLAM depends largely on the accuracy of feature extraction and matching, and in practical application environments, for scenes with complex geometric structures or weak textures, a single point feature may not be able to provide enough geometrical information about the environment; meanwhile, the interference of dynamic objects might lead to unstable feature matching, which in turn affects the robustness and accuracy of the SLAM system. Along these lines, many researchers have introduced line feature and dynamic feature rejection techniques to carry out extensive and in-depth research on them.

2.1. Point-Line Feature-Based SLAM

In vSLAM, the fusion of point-line features can effectively improve the system's ability to adapt to complex environments. Most of the current line-feature-based SLAM frameworks use LSD [12] algorithms for line segment detection and further utilize the line binary descriptor (LBD) [13] to complete line feature description and matching [4,5,14–16]. Compared to the EDLines [17] line features-based vSLAM algorithm [18–20], LSD can achieve sub-pixel detection accuracy without adjusting parameters, and the impact of noise

is relatively small. A typical real-time matching algorithm is PL-SLAM [4,5], which employs LSD and LBD for line segment matching. Pumarola et al. [4] used the length and direction of line features to remove outliers and solve the problem of line segment tracking and matching, while Gomez Ojeda et al. [5] extended this algorithm to closed-loop detection. Fu et al. [14] effectively integrated point, line, and inertial measurement unit (IMU) data in an optimized sliding window based on the state-of-the-art VINS-mono [2], achieving high-precision attitude estimation. Lim et al. [21] introduced a monocular point-line vSLAM system, which effectively improves the matching accuracy by introducing optical flow tracking and filtering short line segments in each frame. Additionally, studies [22,23] improved the accuracy and stability of the camera's pose estimation and depth map recovery in monocular and multi-view SLAM systems by introducing the Manhattan world model to simplify the geometric structure of the environment. During line segment extraction, due to the interference of noise and other uncertainties, the line segment detector may mistakenly detect broken line segments, thereby increasing the difficulty of line feature matching. For this reason, some scholars merged and optimized line features to improve the detection efficiency of algorithms [15,24,25]. In order to reduce time consumption, most of these point-line feature-based methods do not incorporate line segments into loop closure detection, but the inclusion of line features can effectively help robots better perceive and recognize structural features in the scene, especially in weakly textured scenes.

2.2. Dynamic SLAM

Currently, the existing vSLAM solutions in dynamic scenes mainly include multi-sensor information compensation-based methods [26–29], geometric constraint-based methods [10,30–33], and deep learning-based methods [15,34,35]. Among them, representative works such as Kim [28] adopted IMU data to perform rotation compensation on the motion of RGB-D cameras, converting the feature points of the current frame to the coordinate system of the previous frame to generate motion vectors, and then distinguish dynamic feature points in the image. Liu et al. [29] utilized different fields of view of sensors to detect and remove outliers in dynamic scenes. PLDS-SLAM [32] obtained prior features of dynamic regions by detecting and segmenting them, achieving precise separation of dynamic images. Qinglin et al. [33] exploited geometric constraints to classify and filter feature points, and then used the obtained static feature points to complete pose estimation and construct a map. Studies [36,37] utilize optical flow fields with the ability to recognize dynamic features to detect and eliminate motion features.

With the application of deep learning in medical detection [38,39], autonomous driving [40], and 3D point cloud processing[41], some scholars have also introduced different network models into SALM and achieved many significant results. PLD-SLAM [16] applied the MobileNet model and K-Means clustering of depth information to obtain dynamic feature points and lines, and further used epipolar constraints and depth differences to remove dynamic features. Ai et al. [35] introduced multi-view geometric constraints to judge the confidence of the segmented region on the basis of semantic segmentation of images using the DUNet network. In [42,43], the authors drew on depth information to build a depth filter and combined it with semantic information for dynamic object recognition. Among these dynamic SLAM schemes, although the method based on multi-sensor fusion improves system robustness, it also increases hardware cost and computational complexity, and the introduction of sensors increases the uncertainty and noise. Deep learning-based methods can obtain more accurate detection results through object detection or semantic segmentation [44], but most seriously rely on prior knowledge, and model training also requires a lot of time. Hence, in this work, by combining optical flow, geometric constraints, and image depth values, the accuracy of dynamic feature judgment can be effectively improved, and the distinction and recognition of indoor dynamic objects can be realized without the need for semantic segmentation models, as well as other expensive sensors, which not merely effectively reduces the demand for hardware resources, but also achieves satisfactory results in dynamic object detection and rejection.

3. Pipeline

The algorithm in this paper is based on the architecture of vSLAM, which mainly consists of two parts, the front-end and the back-end, as shown in Figure 1. In the front-end, to ensure real-time, the point-line feature separation strategy is used to calculate the LSD line feature descriptor. If there are enough Oriented FAST and Rotated BRIEF (ORB) feature points around the line feature, then the line feature should be removed to reduce the number of line features in the high texture region. Meanwhile, the line features extracted by LSD need to be merged and optimized before feature matching is performed. Then, in terms of the Manhattan world hypothesis, the Manhattan axis is calculated using the structure of line features to optimize the pose estimation of the robot, and the re-projection error of point-line features is weighted to improve the localization accuracy of the system. To reduce the interference of dynamic objects and improve the stability and reliability of the system in indoor dynamic environments, the dynamic point-line features are further identified and rejected by combining LK optical flow, image depth, and geometric constraints, and then the static features with high-confidence are utilized to complete the pose estimation and background reconstruction. In the back-end, on the basis of obtaining the closed-loop candidate keyframe using the BOW model, the structural consistency judgment of line features is further added to increase the accuracy and robustness of loop closure detection through the assistance of line features. Finally, a 3D dense map of the indoor environment is constructed by combining the bundle adjustment (BA) optimized keyframe's pose and the robot's trajectory, and introducing the point cloud library (PCL) and point cloud preprocessing techniques to construct a 3D dense map of the indoor environment, and then using the Octomap to further generate a 3D octree map that can be applied to robot navigation.

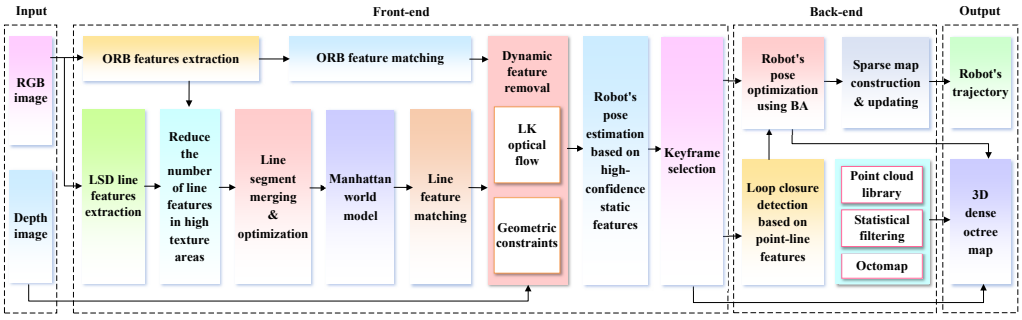


Figure 1. Scheme of visual SLAM for mobile robots based on point-line features and Manhattan world model in indoor dynamic scenes.

3.1. Line Feature Extraction and Optimization

The LSD algorithm can obtain detection results with sub-pixel level accuracy in linear time, but it is difficult to fully match the same line segments extracted by the algorithm in different images, and often detects many short line segments, so the extracted line features need to be optimized. Inspired by the matching method proposed by Zhao et al. [45], we defined attributes such as endpoints, midpoints, and direction vectors of line segments, as well as geometric relationships such as angles and lengths between line segments in our work. Then, we sorted the line segments in descending order by length and constructed a coarse-to-fine judgment method using geometric relationships to determine whether the line segments should be merged or not, so that the set of line features can be optimized for the subsequent feature matching and related analysis tasks.

Specifically, the n line segments extracted by LSD are first sorted in descending order of length to obtain an ordered set of line segments $L = \{l_1, l_2, \cdots, l_n\}$. This processing is mainly because the confidence level of long line segments is higher compared to short line segments, and thus, the short line segments should be merged in the direction of the long line segments. Then, a preliminary judgment is made based on the angle between the line

segments, i.e., coarse selection. Let $s_i = (x_{s_i}, y_{s_i})$, $e_i = (x_{e_i}, y_{e_i})$ and $m_i = (x_{m_i}, y_{m_i})$ denote the start, end, and midpoint of the ith line segment, respectively. Then, the ith line segment can be denoted as $l_i = \overrightarrow{s_i e_i}$, as shown in Figure 2. Assume that the two line segments to be merged are (l_i, l_j), the length of l_i and l_j satisfies $length(l_i) \geq length(l_j)$, and θ_{ij} is the angle between segments l_i and l_j in degrees. If the following condition is satisfied, it indicates that l_i and l_j have preliminarily possessed the merging conditions, i.e.,

$$\theta_{ij} = \arccos\left(\frac{\overrightarrow{s_i e_i} \cdot \overrightarrow{s_j e_j}}{|\overrightarrow{s_i e_i}||\overrightarrow{s_j e_j}|}\right) < \theta_{th} \tag{1}$$

where $|\cdot|$ denotes the modulus of a vector, and θ_{th} is a set angle threshold.

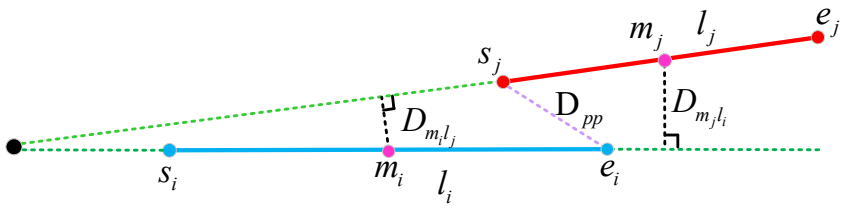


Figure 2. The properties and geometric relationships of two line features extracted by LSD algorithm.

To further determine whether (l_i, l_j) can proceed with the final merger, a fine selection is required based on Equation (1). Thus, the similarity between l_i and l_j is constructed as follows:

$$S(l_i, l_j) = 1 - \frac{1}{\sqrt{2}}\sqrt{\left(\frac{D^*}{\eta\alpha}\right)^2 + \left(\frac{\theta_{ij}}{\eta\theta_{th}}\right)^2} \tag{2}$$

$$\eta = \frac{l_j}{l_i}, \eta \in (0, 1] \tag{3}$$

$$D_{pp} = \min(D_{s_i s_j}, D_{s_i e_j}, D_{e_i e_j}, D_{e_i s_j}) \tag{4}$$

$$D^* = \max(D_{pp}, D_{m_i l_j}, D_{m_j l_i}) \tag{5}$$

where D_{pp} denotes the closest distance between the endpoints of the line segments of l_i and l_j, $D_{m_i l_j}$ and $D_{m_j l_i}$ are the distances from the midpoints l_i and l_j of m_i and m_j to l_j and l_i, respectively, α is the measure, and $\alpha = 1.5$ in this paper. The larger $S(l_i, l_j)$ is, the more similar l_i and l_j are, and the more eligible they are for merging. Here, we believe that $S > 0.3$ satisfies the merging condition of line segments.

According to the above steps, a set of line segments Ξ that can be merged with l_i has been found. During the merge process, it is necessary to project l_j onto l_i, i.e.,

$$l_j = \overrightarrow{s_j e_j} \in \Xi \tag{6}$$

$$l_j{}' = \mathrm{Prj}_{l_i}\overrightarrow{s_j e_j} \tag{7}$$

Let the start and end points of $l_j{}'$ be $s'_j = (x'_{s_j}, y'_{s_j})$ and $e'_j = (x'_{e_j}, y'_{e_j})$, respectively. If $x'_{s_j} > x_{s_i}$ and $x'_{e_j} < x_{e_i}$, it indicates that $l_j{}'$ are all inside l_i, then the merged line segment from l_j to l_i is l_i. If $x'_{e_j} > x_{e_i}$, it indicates that part or all of $l_j{}'$ is outside of l_i, as shown in Figure 3. In this case, considering that the long line segment plays a dominant role in the merging process, to ensure that the merged line segments do not deviate too far from the long line segments, let the starting point of the merged line segment be the endpoint $s_i = (x_{s_i}, y_{s_i})$ of the long line segment, then the merged line segment $l_{j \to i}$ can be obtained according to the following:

$$l_{j\to i} = \overrightarrow{s_i e'_j} + \frac{\eta}{2} \times \overrightarrow{e'_j e_j} \tag{8}$$

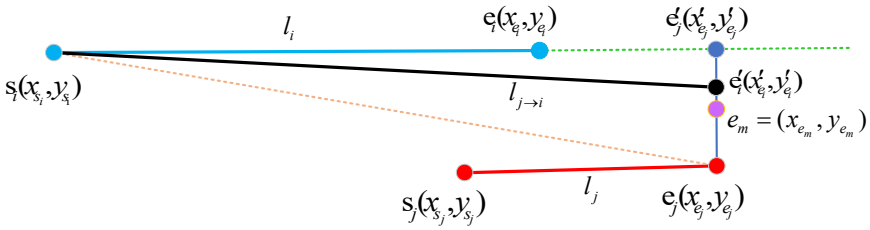
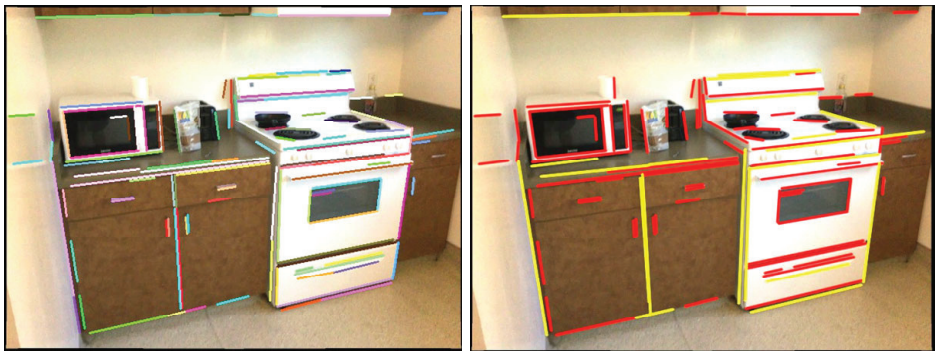


Figure 3. Scenarios for the line features merging.

As a note, the length of line segments is usually closely related to their credibility and merging potential in the merging process. Our method first sorts line segments by length, and then uses the angle between line segments for coarse screening to obtain the set of line segments to be merged, effectively reducing the computational complexity of line segment similarity comparison. Furthermore, this method adaptively adjusts the distance and angle thresholds of the merged line segments based on their length ratio without the need for manual intervention, which can be better adapted to the case of line segments of different lengths, thereby improving the accuracy of merging. Although the long segment is more representative and provides more constraints compared to the short segment, one cannot simply project the short segment onto the long segment. Consequently, the merging distance between short segment l_j and long segment l_i can be adaptively determined under different line segment lengths by comparing the length ratio η of line segments. That is, the closer η is to 0, the greater the difference in length between l_j and l_i, and the closer the endpoint $e'_i = (x'_{e_i}, y'_{e_i})$ of the merged $l_{j\to i}$ is to $e'_j = (x'_{e_j}, y'_{e_j})$. The closer η is to 1, the smaller the length difference between l_j and l_i, and the closer the endpoint $e'_i = (x'_{e_i}, y'_{e_i})$ of the merged $l_{j\to i}$ $e_m = (x_{e_m}, y_{e_m})$. It can be concluded that the effect of different line lengths can be balanced by our method, which contributes to reducing the merging error and improving the accuracy.

Figure 4 compares the line feature extraction effects of different algorithms. Among them, Figure 4a shows the line features extracted from the scene using the traditional LSD algorithm, which are distinguished by different colors, while Figure 4b shows the line features obtained after merging by our method. The line segments that meet the merging conditions are marked in yellow, while those that do not meet the merging conditions are marked in red. We find that the line features obtained after merging and optimizing using our method are significantly better than conventional LSD.



(**a**) LSD (**b**) Ours

Figure 4. Comparison of line feature extraction results for different algorithms.

3.2. Manhattan Axis Calculation

Manhattan world is a universally applicable model for indoor environments and structured scenes, and its key assumption is that elements in the environment (e.g., walls, floors, ceilings, and other key structures) are mainly aligned along three mutually perpendicular axes (i.e., Manhattan axes). Therefore, this paper applies the Manhattan world assumption to the SLAM system, namely, on the basis of utilizing the position estimation of point and line features, further utilizing the additional constraints of structural regularity of the Manhattan world to improve the accuracy of the pose estimation and the consistency of the map reconstruction.

In the process of extracting the Manhattan coordinate system, we mainly use the direction vector of line features in 3D space to project the direction information of line features onto the Manhattan sphere and employ MeanShift clustering to find the clustering centers of the direction in space [22,46]. Then, Manhattan axis selection is achieved by minimizing the difference between the direction vector and the ideal orthogonal direction, i.e.,

$$\min f(\overrightarrow{A_1}, \overrightarrow{A_2}, \overrightarrow{A_3}) = \sum_{i \neq j} \left| \overrightarrow{A_i} \cdot \overrightarrow{A_j} \right| + \lambda \sum_{i=1}^{3} (1 - \left\| \overrightarrow{A_i} \right\|^2) \tag{9}$$

where $\overrightarrow{A_1}$, $\overrightarrow{A_2}$, and $\overrightarrow{A_3}$ denote the closest axis directions to the Manhattan coordinate system determined using MeanShift clustering, respectively; $\sum_{i \neq j} \left| \overrightarrow{A_i} \cdot \overrightarrow{A_j} \right|$ denotes the calculation of the orthogonal product of the two principal axes, ideally, the value equal to 0; $\lambda \sum_{i=1}^{3} (1 - \left\| \overrightarrow{A_i} \right\|^2)$ denotes regularizing the module length to ensure that the module length of each direction vector is close to 1; λ is a regularization coefficient used to balance the proportion of orthogonality and module length standardization. By solving Equation (9), one can find the Manhattan coordinate system.

Further, the results of MeanShift clustering are converted to the unit sphere for normalization through the Riemann exponent, also, to ensure the orthogonality of the rotation matrix, the singular value decomposition is used to obtain the rotation matrix R_{cm}. Finally, the rotation between the Manhattan coordinate system and the camera coordinate system is represented by the rotation matrix R_{cm} as follows:

$$C_w = R_{wc} R_{cm} C_m \tag{10}$$

where C_w is the world coordinates $C_w = (\vec{x}, \vec{y}, \vec{z})$, and R_{wc} is the rotation matrix from the camera coordinate system to the world coordinates.

Figure 5 shows how we calculate the rotation matrix of a robot in the Manhattan world coordinate system. Assuming there are two local Manhattan coordinates M_1 and M_2 in the current scene, which is composed of line feature directions of different indoor object structures and observed at different keyframes. Meanwhile, assuming that M_1 and M_2 were not recorded in the local Manhattan coordinate system maps prior to C_1, and that they were observed earliest by C_1 and C_3, respectively, one can use C_1 as a reference frame and use the Manhattan rotation matrix to estimate the camera rotation matrix R_{c2}^{w} for C_2, i.e.,

$$R_{c2}^{w} = R_{c2}^{m1} (R_{c1}^{m1})^{T} \mathrm{R}_{c1}^{w} \tag{11}$$

Moreover, M_2 was observed for the first time in C_3, which was not recorded in the local Manhattan coordinate system map and cannot be used for camera rotation matrix estimation based on M_2. However, C_2 can be used as a reference frame to estimate the camera rotation matrix R_{c3}^{w} of C_3 based on M_1, i.e.,

$$R_{c3}^{w} = R_{c3}^{m1} (R_{c2}^{m1})^{T} \mathrm{R}_{c2}^{w} \tag{12}$$

Similarly, the camera rotation matrix R_{c4}^{w} of C_4 is estimated by using C_3 as a reference frame according to M_2, i.e.,

$$R_{c4}^{w} = R_{c4}^{m2}(R_{c3}^{m2})^{T}R_{c3}^{w} \tag{13}$$

It can be found that the role of the Manhattan axes in pose estimation is mainly to provide a structured and globally consistent model of the environment, enabling vSLAM in indoor structured environments to enhance the accuracy and robustness of feature matching, and pose optimization, by exploiting line features that are aligned with the three main orthogonal directions (i.e., Manhattan axes). This approach improves the quality of localization and map construction in indoor structured environments by reducing mismatches and optimizing pose estimation for geometric consistency with the environment.

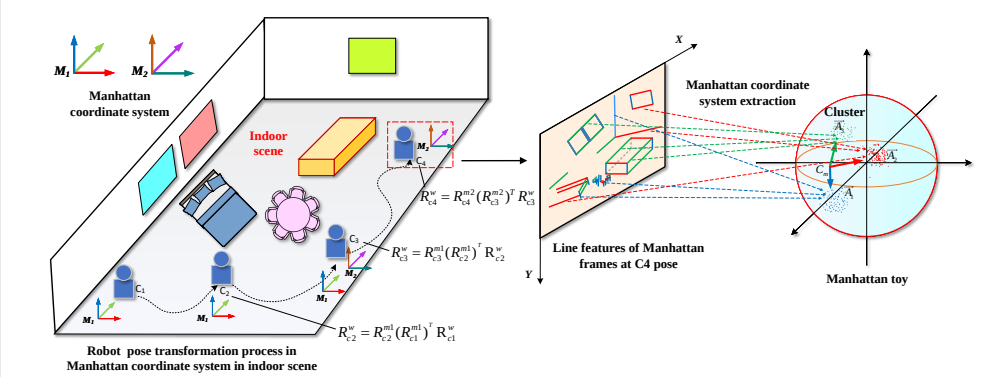


Figure 5. Scenarios for the extraction of Manhattan coordinates in indoor scene.

3.3. High-Precision Matching for Line Feature

For the line feature matching, the common approaches such as Pumarola [4] and Li [47] adopted a similar approach to the processing of point features in ORB-SLAM2, i.e., when searching for matching line segment feature pairs between two frames of images, the descriptors of the line features are directly calculated, and the KNN algorithm is employed to register the line features. Unfortunately, this process heavily relies on the similarity of descriptors, making it easy for mismatches to occur.

In this work, to improve the accuracy and reliability of matching, we use geometric constraints to further validate and screen the matching results based on the similarity of descriptors, and calculate the reprojection error. More specifically, the Hamming distance between descriptors is used to initially screen possible matches to obtain the line feature l_i^- in the previous frame f_{cur}^- that matches the ith line feature l_i in the current frame f_{cur}, which can be computed based on the mapping relation, i.e.,

$$l_i' = F^{\mathrm{T}} l_i^- \tag{14}$$

$$F = K^{-T} t^{\wedge} R K^{-1} \tag{15}$$

where K denotes the camera's inner reference matrix, R and t denote the camera rotation matrix and translation vector, respectively. Ideally, all points on l_i should be on l_i', which is satisfied for any point $p(x, y, 1)$ of l_i, i.e.,

$$p^T l_i' = 0 \tag{16}$$

By substituting Equation (14) into Equation (16), the geometric relationship between the line features of two frames can be obtained as follows:

$$p^T F^T l_i^- = 0 \tag{17}$$

In fact, there may be some error between the line l_i' obtained through fundamental matrix mapping and the actual observed line l_i. Therefore, we calculate the sum of the

distances between the starting point (s_i, s') and the ending point (e_i, e_i') of the two line segments, respectively, to obtain the matching error of the two line features. Then, by setting an error threshold D_{th}, potential mismatches are filtered out to improve the accuracy of the matching, i.e.,

$$\left\|s_i - s'_i\right\| + \left\|e_i - e'_i\right\| \leq D_{th} \tag{18}$$

According to the above analysis, the pseudo-code of the high-precision matching algorithm for inter-frame line features in this paper is shown in Algorithm 1.

Algorithm 1 High-precision matching of line features

Require: line feature set KeyL1 of f_{cur}^{-} and the descriptor mLdes1; line feature set KeyL2 of f_{cur} and the descriptor mLdes2; Foundational matrix F

Ensure: Refined matching list of line features VL

1: $L(l_i, l_i^-) \Leftarrow \text{KnnMatch}(mLdes1, mLdes2)$
2: **for** $i = 1$ to N **do**
3: $\quad l_i' \Leftarrow \text{Projection}(F, l_i^-)$
4: $\quad D(\|s_i - s_i'\|, \|e_i - e_i'\|) \Leftarrow \text{Distance}(l_i', l_i)$
5: $\quad$ **if** $\|s_i - s_i'\| + \|e_i - e_i'\| \leq D_{th}$ **then**
6: $\quad\quad VL \Leftarrow (l_i, l_i^-)$
7: $\quad$ **end if**
8: **end for**

Figure 6 shows the line feature matching results of different methods. It can be observed that directly using descriptors for KNN matching will result in a certain number of matching errors. Instead, the line feature matching using basic matrices with geometric constraints ensures geometric consistency while effectively filtering out mismatches, thereby improving the accuracy of matching.

(**a**) KNN (**b**) KNN and geometric constraints

Figure 6. Comparison of the line feature matching results for different methods.

3.4. Dynamic Feature Detection and Elimination

In dynamic scenes, optical flow can be used to continuously track features and establish corresponding relationships between adjacent frame features to assist in feature matching. Also, effective detection of moving targets can also be achieved by utilizing the characteristics of the changing optical flow of moving targets. Thus, in this paper, a dynamic feature detection method based on LK optical flow and geometric constraints is constructed, and histograms are established by dividing the dynamic point-line features with the depth information of the image. First, the features of the current frame are tracked using the LK optical flow, and the accurate optical flow information is obtained by minimizing the matching error (i.e., optical flow residual) around each feature point. Assuming that the coordinate point to be tracked is (u_x, u_y), the optical flow vector $E(d_x, d_y)$ is calculated by setting the neighborhood window size to $w_x \times w_y$. Then, for each pixel, the optical flow residuals between the predicted position and the actual observed position after the optical flow vector shift are calculated as follows:

$$\Delta \mathrm{d} = \sum_{x=u_x-w_x}^{u_x+w_x} \sum_{y=u_y-w_y}^{u_y+w_y} (f_{\bar{cur}}(x,y) - f_{cur}(x+d_x+\Delta \mathrm{d}_x, y+d_y+\Delta \mathrm{d}_y))^2 \tag{19}$$

After obtaining the optical flow residuals of adjacent frames, a threshold d_{th} is set, and if it satisfies $\Delta d > d_{th}$, it is considered that the object's movement distance in the two frames is relatively large, and it can be judged as a dynamic object. Considering that the conventional optical flow method cannot effectively recognize objects with small movement amplitude, this paper considers introducing geometric constraints on the basis of optical flow and calculating the global similarity through the distance error of point features and line features to further improve the accuracy of dynamic feature detection. Notably, in our work, global similarity is regarded as the consistency of point-line features in geometric space.

For point features, the global similarity is calculated by using epipolar constraints to match the distance between the point and its corresponding epipolar. If the result is greater than a certain threshold, it is marked as a dynamic point. Specifically, according to the fundamental matrix F, map the feature points of $f_{\bar{cur}}$ to the corresponding search area of f_{cur}, i.e., near the epipolar $l_i^{f_{cur}}$, and calculate the distance error M_p as follows:

$$M_p = \frac{\left| {\mathrm{P}_i^{f_{cur}}}^T F P_i^{f_{\bar{cur}}} \right|}{\sqrt{\|A\|^2 + \|B\|^2}} \tag{20}$$

$$l_i^{f_{cur}} = [A,B,C]^T = F P_i^{f_{\bar{cur}}} \tag{21}$$

where $P_i^{f_{\bar{cur}}} = \left[u_i^{f_{\bar{cur}}}, v_i^{f_{\bar{cur}}}, 1\right]$ and $P_i^{f_{cur}} = \left[u_i^{f_{cur}}, v_i^{f_{cur}}, 1\right]$ denote homogeneous coordinate representations of the matching pixels of $f_{\bar{cur}}$ and f_{cur}, respectively, and $[A,B,C]^T$ is the vector representation of the polar line $l_i^{f_{cur}}$.

For line features, the observed line features are constantly changing due to the continuous motion of the robot, while the line features of dynamic objects do not conform to the rigid-body transformation, and the range of spatial distance variations between two consecutive frames of the line features tends to be large. Therefore, we adopt the algorithm proposed by Huang L.M [48], which can simultaneously solve line segment matching and motion estimation and utilize the integral of the distance between the corresponding points of the line features as a measurement function to calculate the global similarity between the line features to determine whether the matched pairs of line features between two consecutive frames are dynamic or not, i.e., calculate the error distance M_l between the two line features as follows:

$$M_l(l_i^{f_{cur}}, l_i^{f_{\bar{cur}}}) = \left| l_i^{f_{cur}} \right| (m_i^{f_{cur}} - R m_i^{f_{\bar{cur}}} - t)^2 + \frac{\left| l_i^{f_{cur}} \right|^3}{12} (v_i^{f_{cur}} - R v_i^{f_{\bar{cur}}})^2 \tag{22}$$

where $(l_i^{f_{cur}}, l_i^{f_{\bar{cur}}})$ denotes the pair of line features in the two frames that match successfully; $|l_i|$, m_i and v_i denote the length, midpoint, and direction of the corresponding line feature, respectively. If $M_l(l_i^{f_{cur}}, l_i^{f_{\bar{cur}}})$ is greater than a set threshold, $l_i^{f_{cur}}$ is labeled as a dynamic line feature.

Additionally, since features on dynamic objects tend to have the same depth, we further combine the dynamic features in f_{cur} detected using optical flow and geometric methods with their depth information to find the minimum and maximum depth values of the pixels in f_{cur} constituting the interval $[d_{\min}^{f_{cur}}, d_{\max}^{f_{cur}}]$, and construct the depth histogram

S. In our case, the constant interval width is set to $\xi = 0.3$, and the depth histogram S is constructed based on the following:

$$\frac{d_{\max}^{f_{cur}} - d_{\min}^{f_{cur}}}{\xi} = n \tag{23}$$

If $n \geq 12$, then the histogram is $S = \{s_1, s_2, \cdots, s_n\}$. If $n < 2$, let $n = 12$, recalculate the value of ξ, and evenly divide the interval based on ξ to obtain the histogram $S = \{s_1, s_2, \cdots, s_{12}\}$. On this basis, the dynamic point-line features are put into the corresponding depth-value interval in the histogram to obtain the depth-value interval that preserves the most dynamic features, and all the point-line features that conform to the depth of this interval are labeled as dynamic features, thereby obtaining a static feature point set with high confidence. Such processing can effectively reduce the trajectory error of the robot and thus improve the performance of the SLAM system in dynamic environments.

3.5. Pose Estimation

In our work, to achieve accurate estimations of poses between adjacent frames of images, we utilize Manhattan structural constraints to achieve estimation without rotational drift. Under the Manhattan world model, by detecting all frames f_{man} in the scene that can be represented by the Manhattan world model, i.e., Manhattan frames, and using f_{man} as a reference frame to solve the pose, the rotation matrix $R_{f_{cur}}^{w}$ of the current frame relative to the reference frame is obtained as follows:

$$R_{f_{cur}}^{w} = R_{f_{cur}}^{m} (R_{f_{ref}}^{m})^{T} R_{f_{ref}}^{w} \tag{24}$$

where $R_{f_{cur}}^{m}$ denotes the Manhattan rotation matrix of the current frame, f_{ref} denotes the reference frame corresponding to the pose solution, $R_{f_{ref}}^{w}$ and $R_{f_{ref}}^{m}$ denote the rotation matrix of the reference frame and the corresponding Manhattan rotation matrix, respectively. The translation t can be found by optimizing the $E\{R_{f_{cur}}^{w}, t\}$ of the minimization error function for point-line feature matching, i.e.,

$$E = \arg\min_{R_{f_{cur}}^{w}, t} \frac{1}{2} \left[\sum_{p_i \in P_{cur}} \psi E_{p_i}^{2} + \sum_{l_i \in L_{cur}} (1 - \psi) E_{l_i}^{2} \right] \tag{25}$$

$$\psi = \frac{N_{P_{cur}}}{N_{P_{cur}} + N_{L_{cur}}} \tag{26}$$

where P_{cur} and L_{cur} denote the sets of point features and line features in f_{cur} that match successfully with f_{ref}, respectively; $N_{P_{cur}}$ and $N_{L_{cur}}$ denote the number of point features and line features in the set, respectively; the reprojection error E_{p_i} of the ith feature point is represented as follows:

$$E_{p_i} = \left\| p_i^{f_{cur}} - \pi(K, R_{f_{cur}}^{w}, t, P_i) \right\|^2 \tag{27}$$

where P_i and $p_i^{f_{cur}}$ denote the ith 3D spatial point and its corresponding pixel coordinates, respectively, and $\pi(\cdot)$ is the projection function from world coordinates to pixel coordinates.

The reprojection error of the ith line feature is expressed as follows:

$$E_{l_i} = \left\| l_i^{f_{cur}} - \pi(K, R_{f_{cur}}^{w}, t, L_i) \right\|^2 \tag{28}$$

where L_i and $l_i^{f_{cur}}$ are the line segments in 3D space and the corresponding line segments in the image plane, respectively. Herein, line features are represented by Plucker coordinates.

Figure 7 illustrates the reprojection error relation for the point-line feature, where L_i and P_i denote line segments and points in 3D space, respectively; $l_i^{f_{ref}}$ and $p_i^{f_{ref}}$ are the line

segments and points corresponding to L_i and P_i on the image of the reference frame f_{ref}, respectively; the black line segment l_i' is the projection of $l_i^{f_{ref}}$ on f_{cur}; d_1 and d_2 denote the distances from the two endpoints of $l_i^{f_{cur}}$ to the line segment l_i', respectively; and p_i' denotes the projection of $p_i^{f_{ref}}$ on f_{cur}.

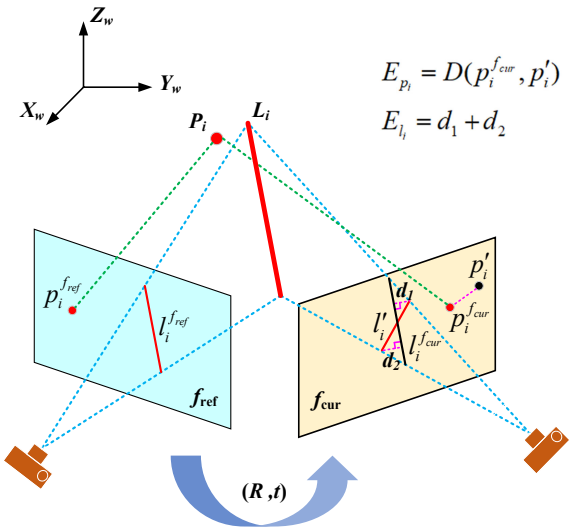


Figure 7. Reprojection error of point-line features.

It should be noted that in environments that do not meet the Manhattan world's assumption or in cases where Manhattan coordinate extraction fails, it is not possible to calculate the camera rotation matrix of the current frame using the Manhattan coordinate system. In this case, we project map points and map lines to the current frame, calculate the motion of two sets of 3D–2D feature pairs, and directly solve the minimum error function through Equation (25) to optimize the camera's pose.

3.6. Loop Closure Detection

To improve the accuracy and robustness of loop closure detection in indoor dynamic scenes, we combine the conventional BOW model with line features to transform the current image into a point-based features visual vocabulary vector and the form of a line-based features set. Then, the set of closed-loop candidate keyframes is obtained on the basis of the similarity score of the BOW vector of point features, and then the Jaccard index values of the line features in the current keyframe and each closed-loop candidate keyframe is compared. If the similarity threshold is satisfied, it means that a closed-loop candidate frame with the test of the line features is obtained at f_{cur}^{key}.

In conventional BOW, the similarity score between two images is compared by calculating the following BOW vector, i.e.,

$$S_{bow}^{i} = 2\sum_{i=1}^{n} |w_i| + |o_i| + |w_i - o_i| \tag{29}$$

where o_i denotes the BOW vector of f_{cur}^{key}, w_i denotes the BOW vector of a covisibility keyframe f_{kef}^{i}. By calculating the similarity score between the point feature BOW vector of f_{kef}^{i} and f_{cur}^{key}, the set I_{cand} of t closed-loop candidate keyframes are obtained, i.e.,

$$I_{cand} = \{f_{cand}^{j} | j \in 1, \cdots, t\} \tag{30}$$

Since the BOW model only considers the number of occurrences of visual words when filtering candidate keyframes, and does not take into account the distribution of visual words in the image and their semantic relevance, it is prone to more false-positive detection results. Therefore, we further utilize the line feature to calculate the Jaccard index value of f_{cur}^{key} with each f_{cand}^{j}, i.e.,

$$S_{Jac} = \frac{\left| A^L_{f_{cur}^{key}} \cap B^L_{f_{cand}^{j}} \right|}{\left| A^L_{f_{cur}^{key}} \right| + \left| B^L_{f_{cand}^{j}} \right| - \left| A^L_{f_{cur}^{key}} \cap B^L_{f_{cand}^{j}} \right|} \tag{31}$$

where $A^L_{f_{cur}}$ denotes the array of line features in f_{cur}^{key}, $B^L_{f_{cand}^{j}}$ denotes the line feature array of the jth closed-loop candidate keyframe f_{cand}^{j} calculated by the BOW model, $\left| A^L_{f_{cur}^{key}} \cap B^L_{f_{cand}^{j}} \right|$ denotes the number of line features successfully matched in the two frames, and S_{Jac} is within the interval $[0, 1]$.

If at least three consecutive frames in I_{cand} satisfy $S_{Jac}(A^L_{f_{cur}^{key}}, B^L_{f_{cand}^{j}}) > \kappa$ (κ is the threshold for dividing the line feature Jaccard index between two frames), it indicates that a closed-loop is detected at f_{cur}^{key} and subsequent global optimization will be performed. Since the line features have better structural properties than the point features, after obtaining the closed-loop candidate keyframes using the BOW model, the geometric consistency verification can be better carried out by calculating the similarity of the line features, which effectively reduces the probability of false positives and improves the accuracy of loop closure detection. The loop closure detection strategy that integrates point-line features is described in Algorithm 2.

Algorithm 2 Loop closure detection incorporating point-line features

Require: Queue of keyframes to be processed F_Queue, ORB vocabulary OV, Keyframe ID of the last loop closure detection KF_id
Ensure: List of candidate keyframes for closed-loop I_{cand}

$f_{cur}^{key} \Leftarrow$ Get_F_Queue(F_Queue)
if $f_{cur}^{key}\_id > (KF\_id + 10)$ **then**
  pKF1 $\Leftarrow$ GetCovKF(f_{cur}^{key})
  minScore $\Leftarrow S_{bow}(f_{cur}^{key}$, pKF1, $OV)$
  $KF2 \Leftarrow$ GetConKF(f_{cur}^{key})
  if $KF2$ is empty **then**
    return 0
  end if
  pKF3 $\Leftarrow$ FindSharKF(f_{cur}^{key}, $KF2$)
  if $pKF3$ is empty **then**
    return 0
  end if
  pKF4 $\Leftarrow S_{bow}(f_{cur}^{key}$, pKF3, minScore, $OV)$
  if $pKF4$ is empty **then**
    return 0
  end if
  pKF5 $\Leftarrow$ SelectBestCovK(f_{cur}^{key}, pKF3, minScore)
  if $pKF5$ is empty **then**
    return 0
  end if
  $I_{cand} \Leftarrow$ JacLScore(f_{cur}^{key}, $pKF5$, $LScore$)
end if

3.7. Map Building

On the basis of obtaining the keyframe's pose in the visual odometry of the front-end, we further combine the RGB and depth information of the keyframe to calculate the 3D point coordinates corresponding to each pixel in the keyframe. Then, we utilize a transformation matrix to convert it to the world coordinate system and splice it with the previous 3D point cloud to obtain the global 3D dense point cloud at the current time. First, traverse all the keyframes according to the pinhole camera model, convert the 2D coordinates of each pixel in the RGB map and depth map to the 3D coordinates corresponding to it under the world coordinate system, and then obtain a set of 3D point clouds. For the ith pixel, the following result holds:

$$z_i[u_i, v_i, 1]^T = KTP_i \tag{32}$$

where $[u_i, v_i, 1]^T$ and z_i denote the homogeneous coordinates and the depth of the ith pixel, respectively, P_i is the 3D point corresponding to the ith pixel in the world coordinate system, and T is a transformation matrix composed of the camera rotation matrix R and the translation vector t.

It should be noted that for each inserted keyframe, we need to traverse each pixel point in the keyframe to eliminate the points with abnormal depth values, and when processing the keyframe pixel information, for the kth depth interval $[s_k^{lower}, s_k^{upper}]$ that has the most dynamic features, if the pixel depth information satisfies the following condition:

$$s_k^{lower} \leq z_i \leq s_k^{upper} \tag{33}$$

then the pixel is no longer involved in the construction of the 3D dense map. By this method, we can effectively build a global consistency map for static background.

Further, a 3D point cloud map can be obtained by projecting the pixel points of the keyframes into the 3D coordinate system for point cloud splicing. To obtain high-quality point clouds, we employ statistical filtering to preprocess the 3D point cloud and model the distance d_{ij} from P_i to k points in its domain using Gaussian distribution. Assuming that the distance between a point P_i and its neighboring points in the point cloud is d_{ij}, $i = 1, 2, ..., m$; $j = 1, 2, ..., k$, the mean and standard deviation of the distance from P_i to all its neighboring points are as follows:

$$\mu = \frac{1}{mk} \sum_{i=1}^{m} \sum_{j=1}^{k} d_{ij} \tag{34}$$

$$\sigma = \sqrt{\frac{1}{mk} \sum_{i=1}^{m} \sum_{j=1}^{k} (d_{ij} - \mu)^2} \tag{35}$$

Traversing each point, only the points whose mean distance falls within the confidence interval of the Gaussian distribution are retained, i.e.,

$$\mu - \tau\sigma < \sum_{j=1}^{k} d_{ij} < \mu + \tau\sigma \tag{36}$$

where τ is the coefficient of standard deviation to control the effect of the standard deviation on the distance threshold. According to Equation (36), outliers introduced by sensor noise, measurement error, or system error can be effectively eliminated.

For line features, we discretize each line segment into a series of points and add them to the 3D dense point cloud. Herein, we discretize the line features by setting the sampling interval δ and the minimum number of sampling points as follows:

$$\mathrm{P}_{l_i} = P_{start} + \frac{i}{N} \cdot \overrightarrow{L} \tag{37}$$

where P_{start} denotes the start point of the line segment, $\overrightarrow{L}$ denotes the direction vector of a line segment, N is the number of sampling points, and $N \geq 5$ in this paper.

After the sampling of the line features is completed, the point features corresponding to them are fused with the 3D dense point cloud to obtain a 3D dense map. This approach integrates line features into the mapping, effectively increasing the consistency of the map. Finally, in order to enable the map to be applied to autonomous navigation, we further transformed the obtained 3D dense map into a 3D dense octree map using Octomap [49]. The detailed process is shown in Figure 8.

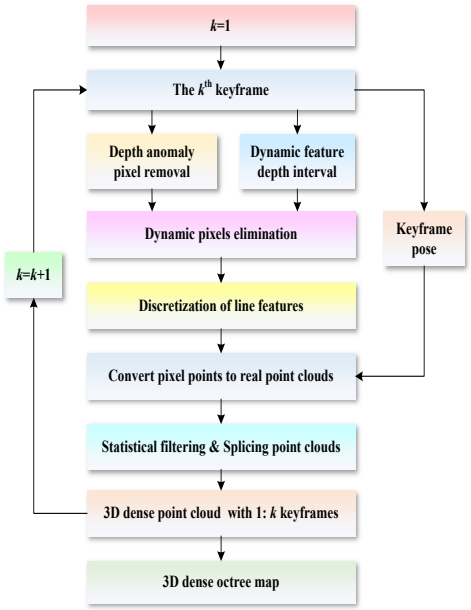


Figure 8. Scenarios for the 3D dense octree map building.

4. Experiments and Results

To verify the feasibility and effectiveness of the proposed algorithm, a series of simulation studies under the TUM dataset and further experimental tests using a mobile robot are given in this section. The TUM dataset [50] is a vSLAM experimental scene dataset created by the Technical University of Munich, which includes RGB-D image sequences and camera movement trajectories in different scenes and is popular within the dynamic vSLAM research community. In this study, all of the experiments were performed on a mobile computer with a 13th Gen Intel(R) Core (TM) i7-13700H 2.40 GHz with 16 GB of RAM and GeForce RTX4060 GPU, running under Ubuntu 18.04.

4.1. Simulation Studies

To evaluate the performance of our method in an indoor dynamic scene, we conducted comparative experiments for ORB-SLAM2 [1], MSC-VO [51], DS-SLAM [52], MR-DVO [53], PL-SLAM [4] and our method on the TUM dataset sequence. In the simulation, we adopted absolute trajectory error (ATE) and relative pose error (RPE) as indicators to measure the accuracy of the algorithm's pose estimation. Among them, ATE intuitively reflects the global positioning accuracy of the algorithm by directly calculating the error between the true and estimated poses of the system, while RPE calculates the difference between the changes in the true and estimated poses of the system at every identical period of time to assess the drift of the system. Then, the statistical value of the overall error of the system is obtained by calculating the root mean square error (RMSE) between the estimated pose and the reference pose at all time points.

The RMSE of the experimental results are shown in Tables 1–3, and all the listed data come from the papers of the corresponding algorithms as well as our experimental results, where "-" indicates that the algorithm did not provide relevant experimental results, and bold indicates the best value on the corresponding dataset sequence. Although some of the algorithms, such as ORB-SLAM2, MSC-VO, and PL-SLAM, have calculated corresponding ATE and RPE; these algorithms were running with tracking failures for part of the time, and thus, these two indicators mainly focus on the error in the successful tracking stage. Moreover, for the monocular PL-SLAM, its initialization time was relatively long and its camera trajectory cannot be obtained in experiments. Therefore, this paper adopted its keyframe trajectory for ATE and RPE calculations, but we believe that it still has a certain reference value.

From Table 1, it can be seen that in high-dynamic scene dataset sequences, our algorithm reduced RSME by 70% compared with ORB-SLAM2, MSC-VO, and PL-SLAM without involving dynamic feature removal, and overall performance was better than MR-DVO using motion methods to remove dynamic features. However, for low-dynamic and static scenes, our algorithm did not show significant advantages, and there was also a gap compared with DS-SLAM based on SegNet deep network. The main reason is that in low dynamic scenes, object motion changes were relatively small, making it difficult to distinguish moving objects using optical flow and geometric methods, and the optical flow method may result in making negative optimization due to camera jitter when calculating the optical flow vectors in low dynamic scenes.

Table 1. Comparison of the RMSE (m) for absolute trajectory error (ATE) among different algorithms on the TUM dataset sequence.

	Dataset Sequence	ORB-SLAM2	MSC-VO	DS-SLAM	MR-DVO	PL-SLAM	Ours
Static	*fr3/long_office*	**0.0149**	0.0444	-	-	1.2005	0.0589
Low-dynamic	*fr3/sitting_static*	0.0085	0.0082	**0.0065**	-	-	0.0133
	fr3/sitting_xyz	**0.0088**	0.0145	-	0.0482	0.3253	0.0686
	fr3/sitting_rpy	0.216	**0.0182**	0.0187	-	-	0.0206
	fr3/sitting_half	0.0312	0.0582	**0.0148**	0.0470	0.3766	0.0254
High-dynamic	*fr3/walking_static*	0.4173	0.1488	**0.0081**	0.0656	-	0.0125
	fr3/walking_xyz	0.8254	0.4870	**0.0247**	0.0932	0.2445	0.0628
	fr3/walking_rpy	0.9269	0.9362	0.4442	**0.1333**	0.1684	0.2925
	fr3/walking_half	0.5459	0.5263	**0.0303**	0.1252	0.4299	0.0650

To demonstrate the superiority of line features in the comparative analysis of RPE, we further added DVO [54] and SLAM [31] to compare relative translation error (RTE) and relative rotation error (RRE). As can be seen from Tables 2 and 3, in high-dynamic scenes, for the methods that do not cull dynamic objects, the rotation and translation errors of the line-feature-based SLAM algorithm were mostly better than those of the point-feature-based ORB-SLAM2 and DVO, which is mainly due to the fact that the line-features have good geometric properties that allow them to be more easily matched with features. In addition, the introduction of the Manhattan world model greatly reduced the impact of pose estimation drift. Thus, in high-dynamic scenes, especially in the $fr3/walking\_rpy$ dataset sequence, the RMSE of our algorithm's RRE was lower than that of all other algorithms, and it was also comparable to the semantic network-based DS-SLAM on other high-dynamic dataset sequences.

Table 2. Comparison of the RMSE for relative translation error (RTE) among different algorithms on the TUM dataset sequence.

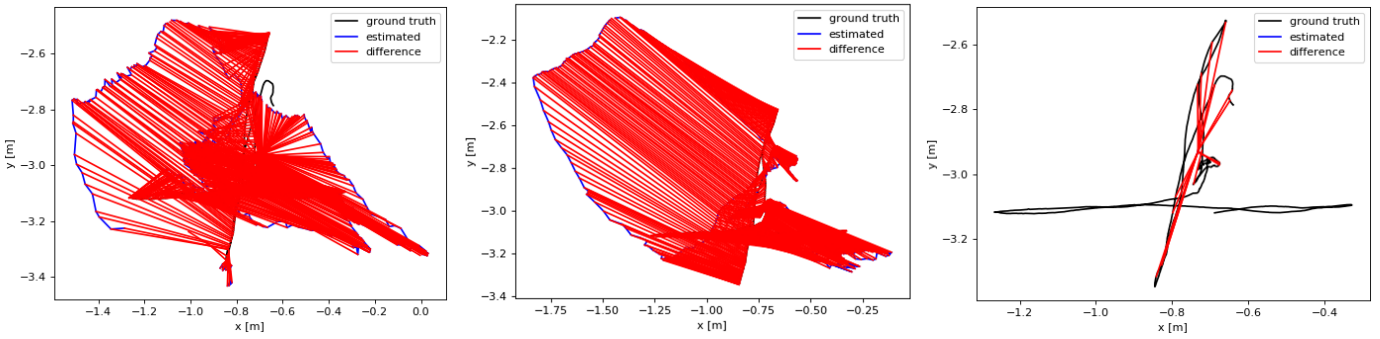
	Dataset Sequence	RMSE (m/s)							
		ORB-SLAM2	MSC-VO	DS-SLAM	MR-DVO	DVO	PL-SLAM	SLAM	Ours
Static	*fr3/long_office*	**0.0081**	0.0095	-	-	0.0231	0.1489	0.0263	0.0129
Low-dynamic	*fr3/sitting_static*	0.0089	0.0098	**0.0078**	-	0.0157	-	0.0174	0.0130
	fr3/sitting_xyz	**0.0110**	0.0170	-	0.0330	0.0453	0.1120	0.0301	0.0351
	fr3/sitting_rpy	0.0247	**0.0216**	-	-	0.1735	-	0.0836	0.0255
	fr3/sitting_half	0.0271	0.0276	**-**	0.0458	0.1005	0.1366	0.0525	**0.0240**
High-dynamic	*fr3/walking_static*	0.2508	0.0917	**0.0102**	0.0842	0.3818	-	0.0234	0.0128
	fr3/walking_xyz	0.4064	0.2630	**0.0333**	0.1214	0.4360	0.2125	0.2433	0.0800
	fr3/walking_rpy	0.4017	0.3058	0.1503	0.1751	0.4038	0.1651	0.1560	**0.1011**
	fr3/walking_half	0.3338	0.2185	**0.0297**	0.1672	0.2628	0.1939	0.1351	0.0215

Table 3. Comparison of the RMSE for relative rotation error (RRE) among different algorithms on the TUM dataset sequence.

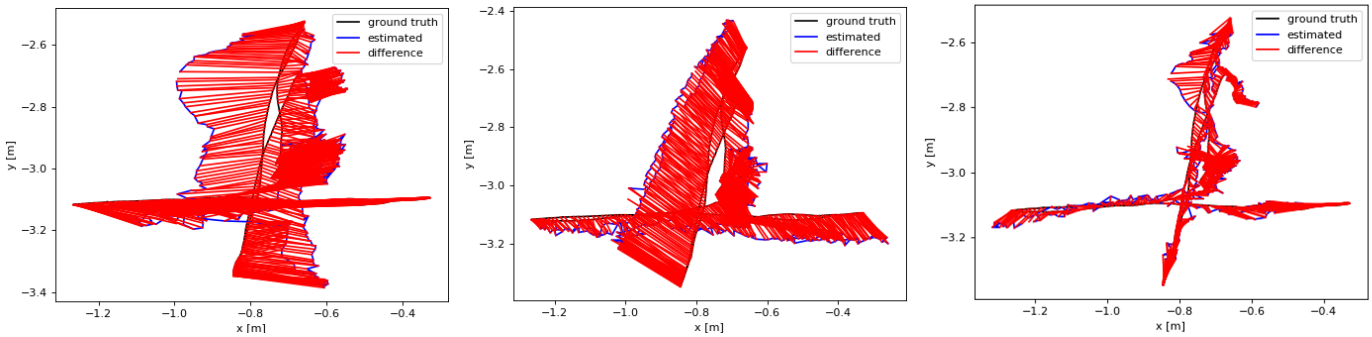
	Dataset Sequence	RMSE (deg/s)							
		ORB-SLAM2	MSC-VO	DS-SLAM	MR-DVO	DVO	PL-SLAM	SLAM	Ours
Static	*fr3/long_office*	0.4638	**0.4418**	-	-	1.5689	1.0345	1.5173	0.4789
Low-dynamic	*fr3/sitting_static*	0.2814	0.2860	**0.2735**	-	0.6084	-	0.7842	0.3501
	fr3/sitting_xyz	**0.4759**	0.5181	-	0.9828	1.4980	0.6182	1.0418	0.7364
	fr3/sitting_rpy	0.7970	**0.6756**	-	-	6.0164	-	5.3861	0.7191
	fr3/sitting_half	0.7363	0.7636	**-**	2.3748	4.6490	**0.4939**	2.8663	0.6848
High-dynamic	*fr3/walking_static*	4.4196	1.8105	**0.2690**	2.0487	6.3502	-	1.8547	0.3098
	fr3/walking_xyz	7.6186	4.8967	**0.8266**	3.2346	7.6669	1.6678	6.9166	1.6627
	fr3/walking_rpy	7.8268	6.1438	3.0042	4.3755	7.0662	**1.7541**	5.5809	2.0076
	fr3/walking_half	6.8215	4.4585	0.8142	5.0108	5.2179	**0.6583**	4.6412	1.3622

For dynamic rejection methods that also utilize point-line features, SLAM [31] ignored the effect of dynamic point features on the whole system, while MR-DVO utilized the foreground model to remove dynamic pixels. In the case of large scene changes, its effect was unsatisfactory. In contrast, the proposed method can effectively adapt to scene changes and detect dynamic objects. Also, the Manhattan world model effectively reduced the error accumulations caused by rotation, thereby improving the accuracy of the robot's pose estimation in indoor dynamic scenes.

Furthermore, for the challenging high-dynamic *walking_xyz* dataset sequence, we utilized the evaluation tool EVO [55] developed by Grupp et al. to display the generated trajectories of different algorithms to evaluate their performance in dynamic scenes. Figures 9 and 10 describe the ATE and visualization trajectories of different algorithms, respectively. It can be observed that owing to the interference of dynamic objects, there was a significant drift in the estimated pose of ORB-SLAM2 and MSC-VO, while PL-SLAM remained in the initialization process due to tracking failure. Instead, the trajectory estimated by our algorithm was more consistent with the real trajectory and maintained stable tracking throughout the entire process, which is mainly due to the fact that our method eliminated dynamic feature points in the front-end and reduced the influence of dynamic objects on the relative positional transformation of the camera during continuous tracking, thus improving the robot's localization accuracy in dynamic scenes.

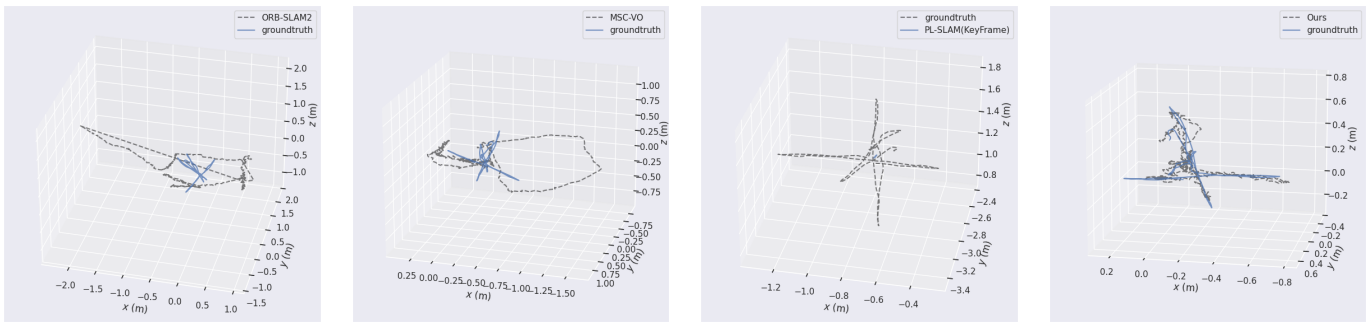


(**a**) ORB-SLAM2 (**b**) MSC-VO (**c**) PL-SLAM



(**d**) Point+optical flow (**e**) Point+line+optical flow (**f**) Point+line+optical flow+geometry

Figure 9. Comparison of the absolute trajectory error (ATE) among different algorithms under the *walking_xyz* dataset sequence.



(**a**) ORB-SLAM2 (**b**) MSC-VO (**c**) PL-SLAM (**d**) Ours

Figure 10. Trajectories estimated by different algorithms under the *walking_xyz* dataset sequence.

Figure 11 further compares the absolute pose error (APE) of different algorithms. Our algorithm exhibited the best global consistency in generating the entire trajectory compared with other testing algorithms. Therefore, we could reasonably conclude that our algorithm possessed smaller errors in various data evaluation indicators in high-dynamic environments and had good robustness.

Figure 12 shows the feature extraction performance of different algorithms on the *walking_xyz* dataset, we find that ORB-SLAM2 and MSC-VO extracted many features on pedestrians, which introduced a lot of error accumulation in subsequent localization and mapping. However, our algorithm can effectively get rid of features on pedestrians, which is due to the merging and optimization of the line features, and in the region where the point features were sufficient, the extracted line features will no longer be tracked. Additionally, we have marked dynamic features and selected dynamic depth intervals in the tracking thread, so the extracted point features on the image were less than the other two algorithms that did not filter out dynamic features. This ensured stable tracking

while reducing the time consumption of feature tracking and BA optimization, and thus improved the real-time performance of the entire system.

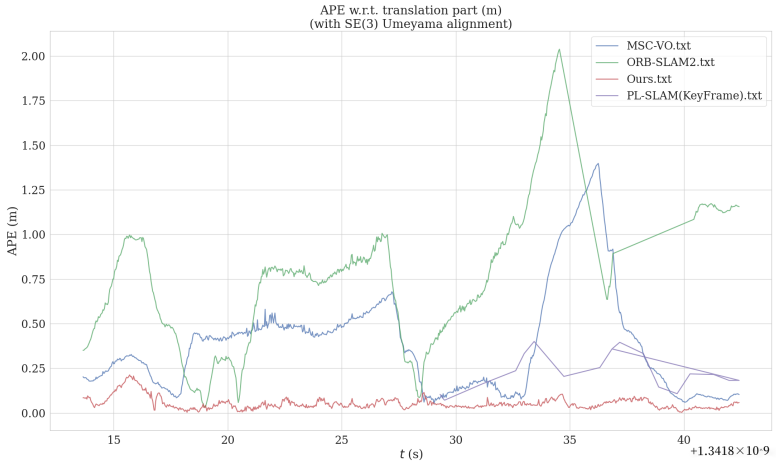
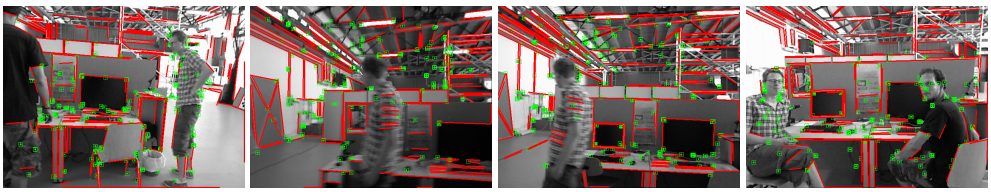
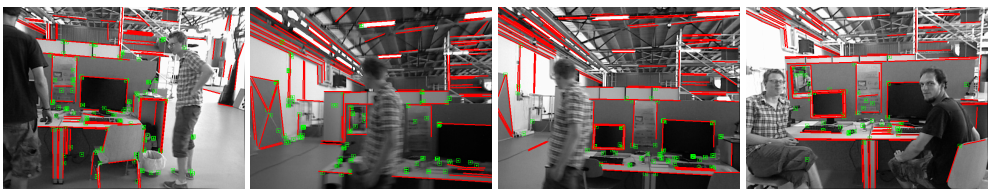


Figure 11. Comparison of the absolute pose error (APE) among different algorithms.

(**a**) ORB-SLAM2



(**b**) MSC-VO



(**c**) Ours

Figure 12. Feature extraction results of different algorithms on the *walking_xyz* dataset sequence.

To verify the mapping effect of the algorithm, we constructed local scene maps on the $fr3/walking\_half\ sphere$ dataset sequence using the three algorithms mentioned above, as shown in Figure 13. It can be seen that ORB-SLAM2 and MSC-VO did not deal with dynamic objects, resulting in a more chaotic map, while the algorithm in this paper can effectively eliminate the point cloud of dynamic objects and realize a clear reconstruction of the real environment.

(**a**) 3D dense map

(**b**) Octree map

Figure 13. Local mapping results of different algorithms under $fr3/walking\_halfsphere$ sequence. From left to right: ORB-SLAM2, MSC-VO, and our algorithm.

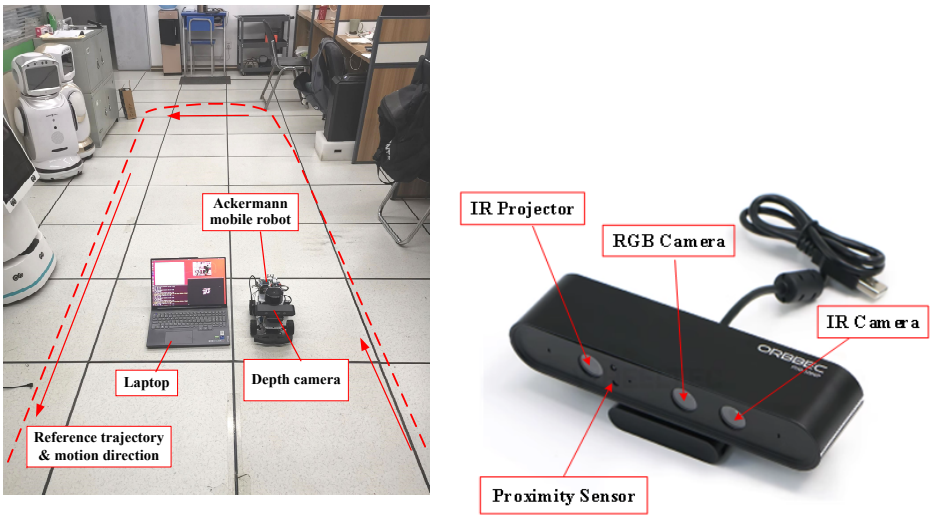
Table 4 analyzes the mean execution time spent on each subtask in the "Tracking" and "Local Mapping" threads of different algorithms on the *walking_xyz* sequence. For the most time-consuming local BA, the performance of our algorithm has improved by 3–6 times after dynamic feature removal. As can be found from the "Tracking" thread, although the introduction of line features improved the accuracy and robustness of the system, it also increased the complexity of the algorithm, thus increasing the overall running time of the system. However, in a high-dynamic dataset sequence, the running time of our algorithm in these two threads was smaller than the other two algorithms, and the efficiency was significantly improved. As a note, ORB-SLAM2 and MSC-VO are prone to tracking loss as they do not have good robustness in dynamic scenes, so we ran multiple times to obtain their mean running time.

Table 4. Comparison of the mean execution time for different algorithms (ms).

Thread	Operation	ORB-SLAM2	MSC-VO	Ours
Local Mapping	KeyFrame Insertion	54.19	23.38	37.76
	Map Features Culling	0.01	0.54	0.45
	Map Features Creation	0.61	2.91	2.81
	Local BA	301.40	164.14	53.51
	KeyFrame Culling	44.685	19.96	12.66
Tracking	Features Extraction	11.84	41.68	43.68
	Initial Pose Estimation	9.47	76.89	20.88
	Track Local Map	2.17	6.72	13.23

4.2. Experimental Testing

To evaluate the performance of the proposed algorithm in real scenes, we conducted experimental tests using a "Wheeltec Ackermann" mobile robot equipped with a "Astra Pro Plus" depth camera in an office with a size of 6.4 m $\times$ 8.4 m, as shown in Figure 14. In the experiment, the robot moved around the room to form a closed loop and a pedestrian was used to walk back and forth within the robot's FOV to simulate unknown dynamic objects in the scene. The mobile robot moved along the reference trajectory indicated by the red dashed line at a speed of 0.2 m/s with a maximum steering speed of 0.3 rad/s. The main parameters of the mobile robot are listed in Table 5.



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

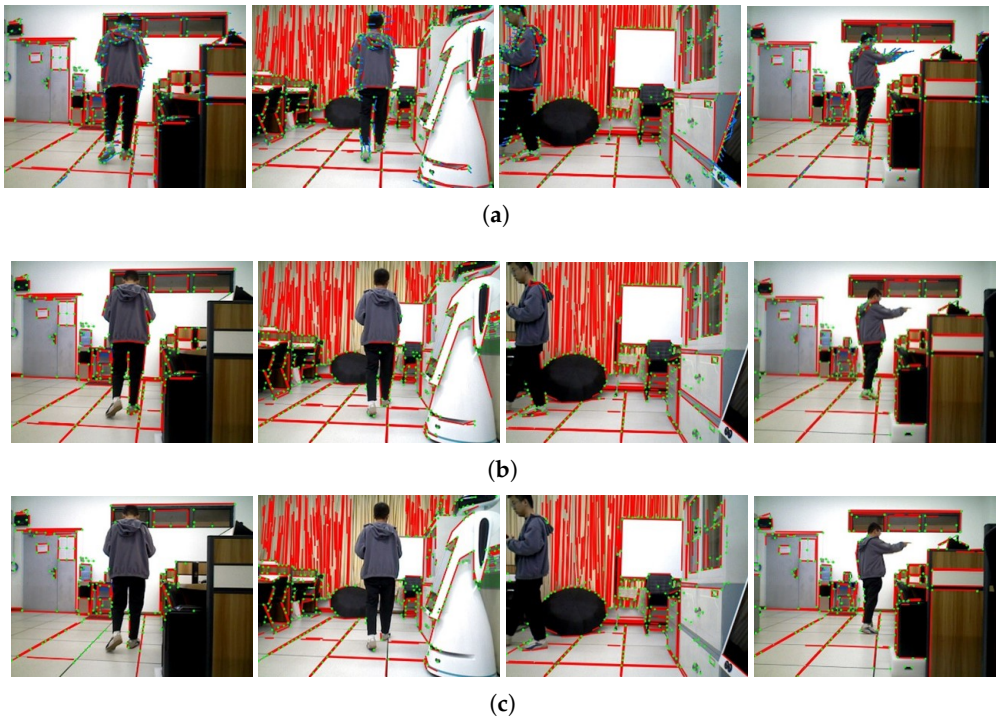
Figure 14. Experimental scene. (**a**) The experimental platform of the mobile robot and the reference trajectory; (**b**) "Astra Pro Plus" depth camera.

Table 5. The main parameters of the "Wheeltec Ackermann" mobile robot.

Parameter	Properties
Size	271 mm × 189 mm × 151 mm
Weight	1.8 kg
Maximum speed	1.2 m/s
Load	3 kg
Main control board	STM32F407VET6

Figure 15 compares the removal effects of dynamic features in real indoor scenes using only sparse optical flow and combining sparse optical flow with geometric methods. In Figure 15a, the green dots indicate the ORB feature points, the red line segments indicate the merged LSD line features, and the blue line segments represent the directions of optical flow. By comparing Figure 15b and Figure 15c, we observe that using only the optical flow cannot accurately identify dynamic feature points in the scene, and cannot effectively detect all dynamic feature points in body parts where pedestrian movements were not obvious. In contrast, we integrated the geometric method on the basis of the optical flow, which can accurately recognize and completely eliminate the dynamic features of the pedestrian.

Figure 16 shows the feature extraction results of different algorithms and the corresponding sparse maps in a real scene. It can be found that the sparse map constructed by the proposed algorithm had better geometry and structure compared to ORB-SLAM2 and was more concise and efficient than MSC-VO. Also, the sparse map points generated on dynamic objects (pedestrians) were significantly less than the other two algorithms after processing the dynamic features in the front end. This further validates that the proposed algorithm effectively improves the real-time performance of the system while ensuring stable tracking.

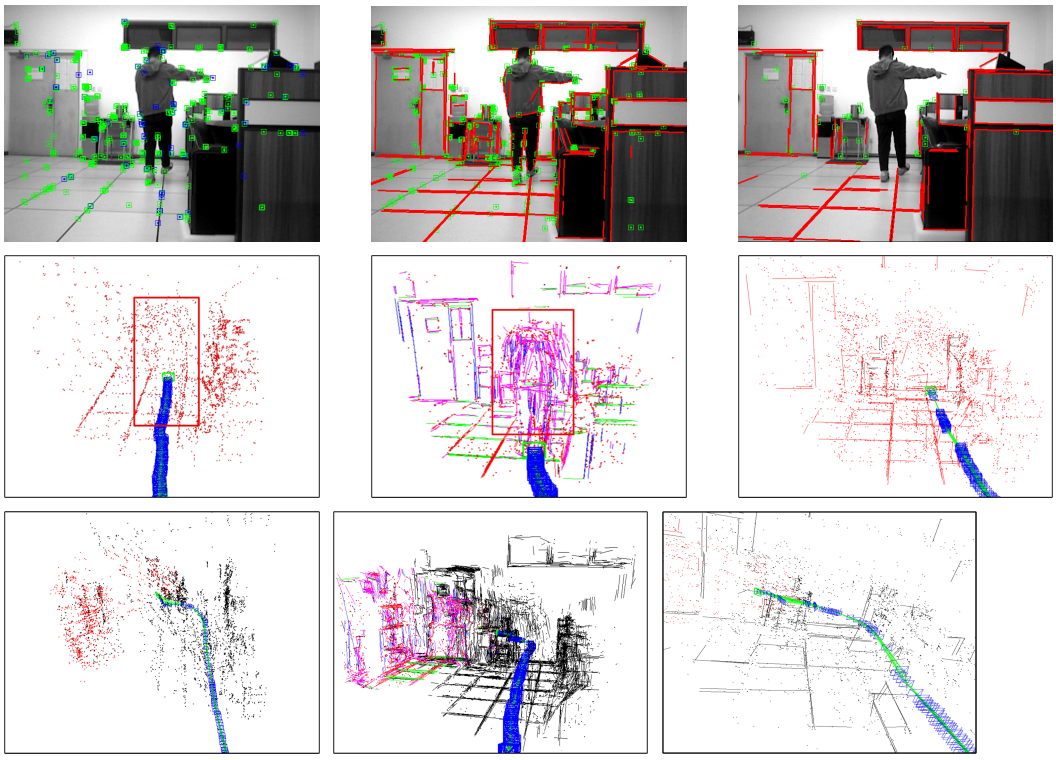


(**a**)

(**b**)

(**c**)

Figure 15. Dynamic feature removal effects. (**a**) Extracted point-line features and optical flow directions; (**b**) dynamic feature culling effect only using sparse optical flow; (**c**) dynamic feature culling effect coupling sparse optical flow with geometric method.



(**a**). ORB-SLAM2 (**b**). MSC-VO (**c**). Ours

Figure 16. The feature extraction results of different algorithms and the corresponding sparse maps in real scenes. The first and second rows show the feature extraction results and the corresponding sparse maps for a certain frame, respectively, and the third row shows the sparse maps after transforming the dynamic features into map points.

Figures 17 and 18 illustrate the trajectory estimation results of different algorithms and their APEs in real dynamic scenes. We find that when the robot moved to the corner, the dynamic target occupied most of the robot's FOV, which makes our algorithm unable to extract sufficiently effective point-line features. This not only failed to leverage the advantages of line features but also matched some incorrect point-line features, resulting in the error in this area being slightly larger than that of ORB-SLAM2. Although MSC-VO also improved the accuracy of pose estimation by extracting point line features and utilizing Manhattan structural constraints, this method was mainly aimed at static scenes, so it failed to detect closed-loop when affected by pedestrian movement back and forth. Conversely, both ORB-SLAM2 and our algorithm can detect closed loops, but because ORB-SLAM2 cannot effectively process dynamic feature points, the robot's pose estimation results were greatly affected when pedestrians wandered within the robot's FOV, which led to the output trajectory having a large error with the reference trajectory in some regions, while our algorithm can effectively identify and remove dynamic feature points, and developed a loop closure detection that integrates point and line features, resulting in a higher degree of fit between the generated trajectory and the reference trajectory.

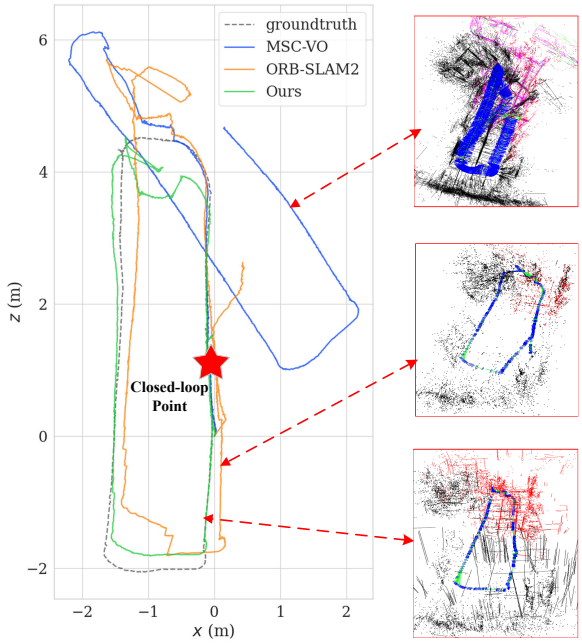


Figure 17. Comparison of trajectories generated by different algorithms in real dynamic scenes.

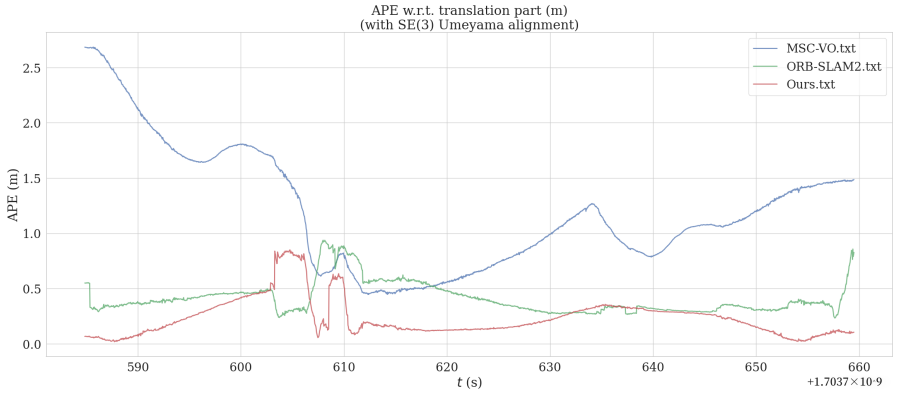


Figure 18. Comparison of the absolute pose error (APE) among different algorithms in real scenes.

To assess the performance of the proposed loop closure detection method, we compared the results of different methods in indoor dynamic scenes and employed the precision–recall curve (P-R curve) as a measurement indicator. Precision represents the proportion of the actual number of closed loops detected by the robot in the loop results, whereas recall represents the proportion of the number of real loops detected by the robot among all real loops. The calculation of accuracy and recall is as follows:

$$P = \frac{TP}{TP + FP} \qquad R = \frac{TP}{TP + FN} \tag{38}$$

where TP and FP denote the true positive and false positive, respectively, and FN denotes the false negative.

As can be seen from Figure 19, the proposed method combined BOW and line features with higher accuracy and recall, and under the same recall rate, it was higher in accuracy than ORB-SLAM2 using only BOW. This is due to the fact that we culled the dynamic features in the front-end, reducing the influence of dynamic objects on the similarity calculation of the BOW model. Moreover, ORB-SLAM2 only used BOW for closed-loop detection. Dynamic object interference would lead to mismatches, which in turn affects the calculation of similarity in BOW. To quantitatively validate the performance of the developed loop closure detection approach in a real scene, we conducted three experiments using the three testing algorithms in the office. Each experiment included three loops and a randomly walking pedestrian, and the tracking failure state considered loop closure detection failures. Although introducing line features increased the running time of the system, the proposed algorithm significantly reduced both the number of keyframes and the tracking time compared with MSC-VO, which does not merely meet the real-time requirements but also ensures stable tracking. The test results are listed in Table 6.

Table 6. Comparison of comprehensive performance for different loop closure detection algorithms.

Algorithm	Number of keyframes	Mean Tracking Time	Number of Closed-Loop	*TP*	*FP*	*FN*	Tracking Failure	Precision	Recall
ORB-SLAM2	481	0.027	3	1	0	1	1	100%	33.3%
MSC-VO	860	0.158	3	0	0	1	2	0	0
Ours	209	0.092	3	3	0	0	0	100%	100%

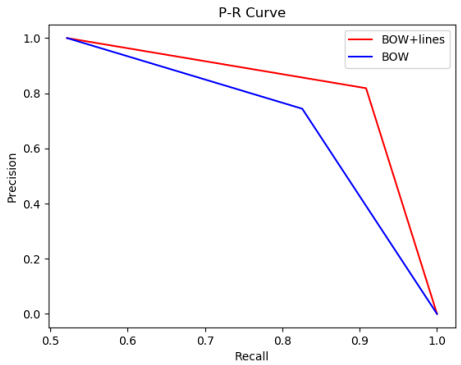
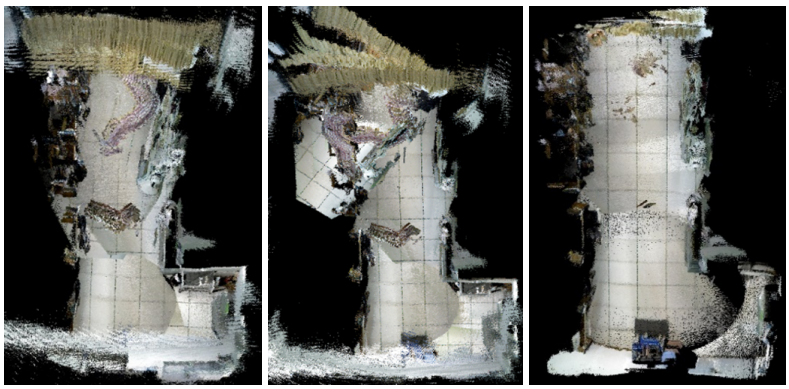


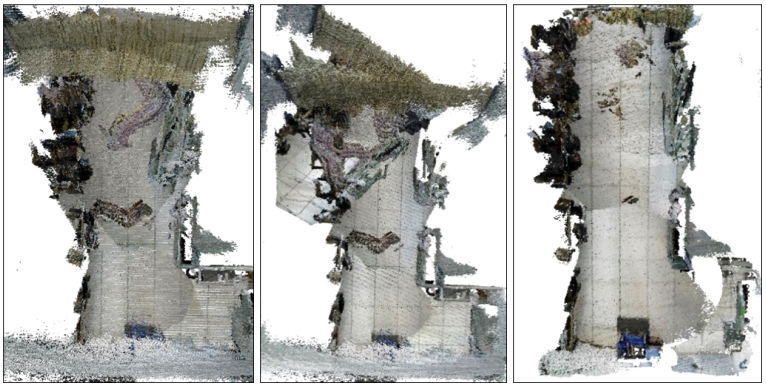
Figure 19. Comparison of the P-R curves for different algorithms.

To demonstrate the global mapping result of the proposed method, we constructed 3D dense maps and corresponding octree maps of the real indoor dynamic scenes, as shown in Figure 20. Apparently, the global map obtained by MSC-VO underwent significant distortion, while the map constructed by ORB-SLAM2 had residual shadows of the pedestrian movement process. Combined with the results in Figure 17, although ORB-SLAM2 can find the correct closed-loop and eliminate the pose drift of the robot to a certain extent, its generated map was distorted and shifted in some regions due to dynamic interference. In contrast, the global map obtained by our algorithm exhibited the best consistency and almost no residual shadow of dynamic objects, which does not merely effectively

avoid the interference caused by dynamic objects but also better reflects the structure of the environment and provides a reliable reference for the autonomous navigation of mobile robots.



(**a**) 3D dense map



(**b**) Octree map

Figure 20. Global mapping results of different algorithms in indoor dynamic scenes. From left to right: ORB-SLAM2, MSC-VO, and our algorithm.

5. Conclusions and Future Work

Aiming at the problem of low localization accuracy and poor consistency of map construction in vSLAM systems in indoor dynamic scenes due to the interference of dynamic objects, this paper proposes a vSLAM algorithm for mobile robots based on point-line features and the Manhattan world model. One of the important works is to complete the extraction and optimization of LSD line features, i.e., the line features are processed mainly for the low-texture region, while the line features are weakened in the feature-rich region, which in turn improves the real-time performance of the system. In order to obtain high-confidence static features to complete the pose estimation and background reconstruction, we combine LK optical flow, image depth, and geometric constraints to identify and reject dynamic point-line features. We also utilize the orientation vectors of the line features to construct a Manhattan world coordinate system, which reduces the cumulative error of rotational estimation during the process of pose estimation. In addition, on the basis of obtaining the closed-loop candidate keyframe using the BOW model, we further add the structural consistency judgment of line features to increase the accuracy and robustness of loop closure detection through the assistance of line features. Simulation results on a public dataset show that line features are more stable than point features, and compared with existing SLAM algorithms based on ORB point features or point-line fusion, this proposed algorithm reduces the RTE and RRE in indoor dynamic environments and improves the accuracy by at least 40%. In terms of dynamic feature processing, the combination of optical flow, limiting geometric constraints and depth can effectively reject features on dynamic

objects; compared with ORB-SLAM2 and MSC-VO, the RSME of ATE in dynamic scenes is reduced by 70% in our algorithm, improving the accuracy of the pose estimation.

In future work, we intend to combine optical flow networks with line features and use light-weight networks to improve the dynamic feature removal effect, thereby making pose estimation more accurate. Moreover, we also plan to deeply combine line features with Manhattan model optimization to better handle robot movement and rotation. Although processing line features may reduce the real-time performance of the SLAM system, it provides more constraints that can greatly improve the robustness and accuracy of the system. Thus, another perspective of this research is to further consider how to coordinate the optimization of point-line features in indoor scenes to improve the real-time performance of the system.

Author Contributions: J.L. (Jiale Liu): formal analysis, investigation, data curation, writing—original draft preparation; J.L. (Jingwen Luo): conceptualization, supervision, project administration, funding acquisition, writing—review and editing. All authors have read and agreed to the published version of the manuscript.

Funding: This work was supported by the National Nature Science Foundation of China under Grants 62063036 and 62463033, and 'Xingdian Talent Support Program' Youth Talent Special Project of Yunnan Province under Grant 01000208019916008.

Institutional Review Board Statement: Not applicable.

Informed Consent Statement: Not applicable.

Data Availability Statement: The raw data supporting the conclusions of this article will be made available by the authors on request.

Conflicts of Interest: The authors declare no conflicts of interest.

Abbreviations

The following abbreviations are used in this manuscript:

PLM-SLAM	The proposed algorithm
vSLAM	Visual Simultaneous Localization And Mapping
LSD	Line Segment Detector
LK	Lucas–Kanade
FOV	Field Of View
BOW	Bag of Words
LBD	Line Binary Descriptor
IMU	Inertial Measurement Unit
BA	Bundle Adjustment
PCL	Point Cloud Library
APE	Absolute Pose Error
ATE	Absolute Trajectory Error
RPE	Relative Pose Error
RMSE	Root Mean Square Error

References

1. 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]
2. 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]
3. Baker, L.; Ventura, J.; Langlotz, T.; Gul, S.; Mills, S.; Zollmann, S. Localization and tracking of stationary users for augmented reality. *Vis. Comput.* **2024**, *40*, 227–244. [CrossRef]
4. Pumarola, A.; Vakhitov, A.; Agudo, A.; Sanfeliu, A.; Moreno-Noguer, F. PL-SLAM: Real-time monocular visual SLAM with points and lines. In Proceedings of the 2017 IEEE International Conference on Robotics and Automation (ICRA), Singapore, 29 May–3 June 2017; pp. 4503–4508. [CrossRef]
5. Gomez-Ojeda, R.; Moreno, F.A.; Zuñiga-Noël, D.; Scaramuzza, D.; Gonzalez-Jimenez, J. PL-SLAM: A Stereo SLAM System Through the Combination of Points and Line Segments. *IEEE Trans. Robot.* **2019**, *35*, 734–746. [CrossRef]

6. Zhang, G.; Lee, J.H.; Lim, J.; Suh, I.H. Building a 3-D Line-Based Map Using a Stereo SLAM. *IEEE Trans. Robot.* **2015**, *31*, 1–14. [CrossRef]
7. Wei, Y.; Zhou, B.; Duan, Y.; Liu, J.; An, D. DO-SLAM: Research and application of semantic SLAM system towards dynamic environments based on object detection. *Appl. Intell.* **2023**, *53*, 30009–30026. [CrossRef]
8. Gong, H.; Gong, L.; Ma, T.; Sun, Z.; Li, L. AHY-SLAM: Toward faster and more accurate visual SLAM in dynamic scenes using homogenized feature extraction and object detection method. *Sensors* **2023**, *23*, 4241. [CrossRef]
9. Islam, Q.U.; Ibrahim, H.; Chin, P.K.; Lim, K.; Abdullah, M.Z.; Khozaei, F. ARD-SLAM: Accurate and robust dynamic SLAM using dynamic object identification and improved multi-view geometrical approaches. *Displays* **2024**, *82*, 102654. [CrossRef]
10. Dai, W.; Zhang, Y.; Li, P.; Fang, Z.; Scherer, S. Rgb-d slam in dynamic environments using point correlations. *IEEE Trans. Pattern Anal. Mach. Intell.* **2020**, *44*, 373–389. [CrossRef] [PubMed]
11. Long, F.; Ding, L.; Li, J. DGFlow-SLAM: A Novel Dynamic Environment RGB-D SLAM without Prior Semantic Knowledge Based on Grid Segmentation of Scene Flow. *Biomimetics* **2022**, *7*, 163. [CrossRef] [PubMed]
12. Von Gioi, R.G.; Jakubowicz, J.; Morel, J.M.; Randall, G. LSD: A fast line segment detector with a false detection control. *IEEE Trans. Pattern Anal. Mach. Intell.* **2008**, *32*, 722–732. [CrossRef]
13. Zhang, L.; Koch, R. An efficient and robust line segment matching approach based on LBD descriptor and pairwise geometric consistency. *J. Vis. Commun. Image Represent.* **2013**, *24*, 794–805. [CrossRef]
14. Fu, Q.; Wang, J.; Yu, H.; Ali, I.; Guo, F.; He, Y.; Zhang, H. PL-VINS: Real-time monocular visual-inertial SLAM with point and line features. *arXiv* **2020**, arXiv:2009.07462.
15. Zhou, F.; Zhang, L.; Deng, C.; Fan, X. Improved point-line feature based visual SLAM method for complex environments. *Sensors* **2021**, *21*, 4604. [CrossRef]
16. Zhang, C.; Huang, T.; Zhang, R.; Yi, X. PLD-SLAM: A new RGB-D SLAM method with point and line features for indoor dynamic scene. *ISPRS Int. J. Geo-Inf.* **2021**, *10*, 163. [CrossRef]
17. Akinlar, C.; Topal, C. EDLines: A real-time line segment detector with a false detection control. *Pattern Recognit. Lett.* **2011**, *32*, 1633–1642. [CrossRef]
18. Zhu, Y.; Jin, R.; Lou, T.-s.; Zhao, L. PLD-VINS: RGBD visual-inertial SLAM with point and line features. *Aerosp. Sci. Technol.* **2021**, *119*, 107185. [CrossRef]
19. Teng, Z.; Han, B.; Cao, J.; Hao, Q.; Tang, X.; Li, Z. PLI-SLAM: A Tightly-Coupled Stereo Visual-Inertial SLAM System with Point and Line Features. *Remote Sens.* **2023**, *15*, 4678. [CrossRef]
20. Ma, X.; Ning, S. Real-Time Visual-Inertial SLAM with Point-Line Feature using Improved EDLines Algorithm. In Proceedings of the 2020 IEEE 5th Information Technology and Mechatronics Engineering Conference (ITOEC), Chongqing, China, 12–14 June 2020; pp. 1323–1327. [CrossRef]
21. Lim, H.; Kim, Y.; Jung, K.; Hu, S.; Myung, H. Avoiding Degeneracy for Monocular Visual SLAM with Point and Line Features. In Proceedings of the 2021 IEEE International Conference on Robotics and Automation (ICRA), Xi'an, China, 30 May–5 June 2021; pp. 11675–11681. [CrossRef]
22. Li, Y.; Brasch, N.; Wang, Y.; Navab, N.; Tombari, F. Structure-SLAM: Low-Drift Monocular SLAM in Indoor Environments. *IEEE Robot. Autom. Lett.* **2020**, *5*, 6583–6590. [CrossRef]
23. Yunus, R.; Li, Y.; Tombari, F. Manhattanslam: Robust planar tracking and mapping leveraging mixture of manhattan frames. In Proceedings of the 2021 IEEE International Conference on Robotics and Automation (ICRA), Xi'an, China, 30 May–5 June 2021; pp. 6687–6693.
24. Hamid, N.; Khan, N. LSM: Perceptually accurate line segment merging. *J. Electron. Imaging* **2016**, *25*, 061620. [CrossRef]
25. Zuo, X.; Xie, X.; Liu, Y.; Huang, G. Robust visual SLAM with point and line features. In Proceedings of the 2017 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS), Vancouver, BC, Canada, 24–28 September 2017; pp. 1775–1782. [CrossRef]
26. Yao, E.; Zhang, H.; Song, H.; Zhang, G. Fast and robust visual odometry with a low-cost IMU in dynamic environments. *Ind. Robot. Int. J. Robot. Res. Appl.* **2019**, *46*, 882–894. [CrossRef]
27. Yang, D.; Bi, S.; Wang, W.; Yuan, C.; Qi, X.; Cai, Y. DRE-SLAM: Dynamic RGB-D encoder SLAM for a differential-drive robot. *Remote Sens.* **2019**, *11*, 380. [CrossRef]
28. Kim, D.H.; Han, S.B.; Kim, J.H. Visual odometry algorithm using an RGB-D sensor and IMU in a highly dynamic environment. In *Robot Intelligence Technology and Applications 3: Results from the 3rd International Conference on Robot Intelligence Technology and Applications*; Springer: Berlin/Heidelberg, Germany, 2015; pp. 11–26.
29. Liu, G.; Zeng, W.; Feng, B.; Xu, F. DMS-SLAM: A general visual SLAM system for dynamic scenes with multiple sensors. *Sensors* **2019**, *19*, 3714. [CrossRef]
30. Sun, Y.; Liu, M.; Meng, M.Q.H. Motion removal for reliable RGB-D SLAM in dynamic environments. *Robot. Auton. Syst.* **2018**, *108*, 115–128. [CrossRef]
31. Zhang, H.; Fang, Z.; Yang, G. RGB-D Visual Odometry in Dynamic Environments Using Line Features. *Robot* **2019**, *41*, 75–82.
32. Yuan, C.; Xu, Y.; Zhou, Q. PLDS-SLAM: Point and line features SLAM in dynamic environment. *Remote Sens.* **2023**, *15*, 1893. [CrossRef]
33. Ai, Q.; Liu, Q.; Xu, Q. An RGB-D SLAM Algorithm for Robot Based on the Improved Geometric andMotion Constraints in Dynamic Environment. *Robot* **2021**, *43*, 167–176.

34. Zhao, X.; Wang, C.; Ang, M.H. Real-time visual-inertial localization using semantic segmentation towards dynamic environments. *IEEE Access* **2020**, *8*, 155047–155059. [CrossRef]
35. Ai, Y.; Rui, T.; Lu, M.; Fu, L.; Liu, S.; Wang, S. DDL-SLAM: A robust RGB-D SLAM in dynamic environments combined with deep learning. *IEEE Access* **2020**, *8*, 162335–162342. [CrossRef]
36. Shimamura, J.; Morimoto, M.; Koike, H. Robust vSLAM for Dynamic Scenes. In Proceedings of the MVA, Nara, Japan, 13–15 June 2011; pp. 344–347.
37. Cheng, J.; Sun, Y.; Meng, M.Q.H. Improving monocular visual SLAM in dynamic environments: An optical-flow-based approach. *Adv. Robot.* **2019**, *33*, 576–589. [CrossRef]
38. Dai, L.; Sheng, B.; Chen, T.; Wu, Q.; Liu, R.; Cai, C.; Wu, L.; Yang, D.; Hamzah, H.; Liu, Y.; et al. A deep learning system for predicting time to progression of diabetic retinopathy. *Nat. Med.* **2024**, *30*, 584–594. [CrossRef] [PubMed]
39. Dai, L.; Wu, L.; Li, H.; Cai, C.; Wu, Q.; Kong, H.; Liu, R.; Wang, X.; Hou, X.; Liu, Y.; et al. A deep learning system for detecting diabetic retinopathy across the disease spectrum. *Nat. Commun.* **2021**, *12*, 3242. [CrossRef] [PubMed]
40. Cui, S.; Liu, F.; Wang, Z.; Zhou, X.; Yang, B.; Li, H.; Yang, J. DAN-YOLO: A Lightweight and Accurate Object Detector Using Dilated Aggregation Network for Autonomous Driving. *Electronics* **2024**, *13*, 3410. [CrossRef]
41. Liu, Y.; Sun, X.; Shao, W.; Yuan, Y. S2ANet: Combining local spectral and spatial point grouping for point cloud processing. *Virtual Real. Intell. Hardw.* **2024**, *6*, 267–279. [CrossRef]
42. Li, S.; Lee, D. RGB-D SLAM in dynamic environments using static point weighting. *IEEE Robot. Autom. Lett.* **2017**, *2*, 2263–2270. [CrossRef]
43. Cui, L.; Ma, C. SDF-SLAM: Semantic depth filter SLAM for dynamic environments. *IEEE Access* **2020**, *8*, 95301–95311. [CrossRef]
44. Li, Y.; Li, Z.; Liu, H.; Wang, Q. ZMNet: Feature fusion and semantic boundary supervision for real-time semantic segmentation. *Vis. Comput.* **2024**, 1–12. [CrossRef]
45. Zhao, Y.; Zhang, H.; Lu, P.; Li, P.; Wu, E.; Sheng, B. DSD-MatchingNet: Deformable sparse-to-dense feature matching for learning accurate correspondences. *Virtual Real. Intell. Hardw.* **2022**, *4*, 432–443. [CrossRef]
46. Peng, X.; Liu, Z.; Wang, Q.; Kim, Y.T.; Lee, H.S. Accurate Visual-Inertial SLAM by Manhattan Frame Re-identification. In Proceedings of the 2021 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS), Prague, Czech Republic, 27 September–October 2021; pp. 5418–5424. [CrossRef]
47. Li, Y.; Yunus, R.; Brasch, N.; Navab, N.; Tombari, F. RGB-D SLAM with structural regularities. In Proceedings of the 2021 IEEE international conference on Robotics and automation (ICRA), Xi'an, China, 30 May 2021–5 June 2021; pp. 11581–11587.
48. Huang, L.; Peng, L.; Cheng, M. Matching and estimating motion of line model using geometric algebra. *J. Image Graph.* **2001**, *6*, 270–274.
49. Hornung, A.; Wurm, K.M.; Bennewitz, M.; Stachniss, C.; Burgard, W. OctoMap: An efficient probabilistic 3D mapping framework based on octrees. *Auton. Robot.* **2013**, *34*, 189–206. [CrossRef]
50. Sturm, J.; Engelhard, N.; Endres, F.; Burgard, W.; Cremers, D. A benchmark for the evaluation of RGB-D SLAM systems. In Proceedings of the 2012 IEEE/RSJ International Conference on Intelligent Robots and Systems, Vilamoura-Algarve, Portugal, 7–12 October 2012; pp. 573–580.
51. Company-Corcoles, J.P.; Garcia-Fidalgo, E.; Ortiz, A. MSC-VO: Exploiting manhattan and structural constraints for visual odometry. *IEEE Robot. Autom. Lett.* **2022**, *7*, 2803–2810. [CrossRef]
52. Yu, C.; Liu, Z.; Liu, X.J.; Xie, F.; Yang, Y.; Wei, Q.; Fei, Q. DS-SLAM: A semantic visual SLAM towards dynamic environments. In Proceedings of the 2018 IEEE/RSJ international conference on intelligent robots and systems (IROS), Madrid, Spain, 1–5 October 2018; pp. 1168–1174.
53. Sun, Y.; Liu, M.; Meng, M.Q.H. Improving RGB-D SLAM in dynamic environments: A motion removal approach. *Robot. Auton. Syst.* **2017**, *89*, 110–122. [CrossRef]
54. Kerl, C.; Sturm, J.; Cremers, D. Robust odometry estimation for RGB-D cameras. In Proceedings of the 2013 IEEE International Conference on Robotics and Automation, Karlsruhe, Germany, 6–10 May 2013; pp. 3748–3754.
55. Grupp, M. EVO: Python Package for the Evaluation of Odometry and SLAM. 2017. Available online: https://github.com/MichaelGrupp/evo (accessed on 3 July 2024).