IEEE Access
Multidisciplinary | Rapid Review | Open Access Journal

Received November 1, 2020, accepted November 29, 2020, date of publication January 6, 2021, date of current version January 15, 2021.

Digital Object Identifier 10.1109/ACCESS.2021.3049467

A SLAM System Based on RGBD Image and Point-Line Feature

DAN LI, SHUANG LIU, WEILAI XIANG, QIWEI TAN, KAICHENG YUAN, ZHEN ZHANG, AND YINGSONG HU

School of Computer Science and Technology, Huazhong University of Science and Technology, Wuhan 430074, China

Corresponding author: Weilai Xiang (m202073320@hust.edu.cn)

This work was supported in part by the National Natural Science Foundation of China under Grant 61502185, and in part by the Fundamental Research Funds for the Central Universities under Grant 2017KFYXJJ071.

ABSTRACT Most of the existing visual SLAM schemes rely solely on point or line features to estimate camera trajectory. In some scenes such as texture missing or motion blurring, it's difficult to find a sufficient number of reliable features, resulting in low positioning accuracy. To extract more features, an RPL-SLAM solution is proposed to extract point features and line features respectively. Ulteriorly, the depth information of RGBD image is used to restore the 3D information of point and line features, improving the accuracy of camera track positioning. RPL-SLAM scheme mainly includes three modules: tracking, local mapping and loop detection. Tracking module extends the application of line feature on the basis of point feature extraction and matching. A SLD line segment extraction algorithm which can eliminate the micro segments and a DBM segment matching algorithm based on word bag are proposed respectively. These two algorithms improve the matching efficiency while ensuring the matching accuracy, and effectively track and locate the camera of each frame. In the local mapping and loop detection module, the Plucker coordinates are applied to express the spatial line and define the re-projection error of the straight line, so that the back-end optimization error model of point-line fusion is unified to solve the instability problem in optimization. RPL-SLAM is tested on TUM RGBD and ICL-NUIM data set respectively, and compared with ORB-SLAM2. The result shows that RPL-SLAM can effectively improve the accuracy of pose estimation and map reconstruction while maintaining real-time performance by fusing point-line features with depth images.

INDEX TERMS SLAM, point-line fusion, Plucker coordinates, tracking, back-end optimization.

I. INTRODUCTION

With the rapid development of augmented reality, drones, robots and other fields in recent years, the technology of Simultaneous Localization And Mapping (SLAM) has also received extensive attention. SLAM solves the problem of how mobile robots or other digital devices progressively establish a consistent map of the surrounding environment while determining its exact location when it is placed in an unknown location in a strange environment [1].

In recent years, several different types of SLAM schemes have been put forward, such as the ORB-SLAM [2] based on the characterization method proposed by Mur-Artal and its ORB-SLAM2 [3] extended to the binocular and depth camera versions, LSD-SLAM [4] based on the direct method proposed by Engel, SVO [5] based on the semi-direct method proposed by Forster. Each of these methods has its characteristics and is suitable for different application scenarios. Among them, the positioning of ORB-SLAM2 performs is more accurate. However, because ORB-SLAM2 applies ORB point features, this method is easy to fail and lacks stability when handling missing texture or textureless scenes, or when feature points temporarily disappear due to motion blurring and other factors. The main reason for the lack of stability is the lack of reliable feature points in the environment in these cases. However, if the advanced features such as lines and planes are considered while processing the features of the points, the stability of the SLAM may be improved.

The line features are very abundant in outdoor environments and are insensitive to changes in illumination compared to point features. Line features can provide more important information about the geometric content of an image, and are more advantageous in a structured environment than point features to represent the environment.

The associate editor coordinating the review of this manuscript and approving it for publication was Heng Wang.

Combining point features and line features can express the environment better and provide rich visual information, making the visual SLAM more robust. However, the application of line features has specific difficulties. Firstly, the extraction and matching of line segments have some problems, such as inaccurate endpoint positions, no unambiguous strong geometric constraints (such as short baseline matching, line endpoint epipolar constraint, etc.), lack of rich textures in line local neighborhood, etc. The second question is the expression and parameterization of the spatial line segment. The existing expression of spatial straight line generally adopts the representation related to the endpoint. However, the method cannot ensure that the end point of the line segment can be stably extracted and observed during the running process. Moreover, over-parameterization problems often exist when the straight line is optimized at the back end, which leads to instability of the optimization solution. Furthermore, when the pose is calculated according to the correspondence of the line segments, the reliability of the line segment usage is further affected because there may be partial occlusion. These problems make the application of line segments in SLAM full of challenges.

In addition, with the popularity of depth cameras, the depth information of pixel points in the collected images is easier to be acquired, so that color images combined with depth images for navigation and positioning calculations are considered to be one of the potential solutions in the future [6]. Therefore, we propose a RPL-SLAM (RGBD and Point -Line SLAM) solution that utilizes depth cameras and is based on point-line feature fusion. The scheme is reconstructed and optimized based on the ORB-SLAM2 framework. At the same time, due to the depth camera, the depth of the line segment endpoint can be directly read from the depth map when the line segment is calculated. Subsequent operations are performed if and only if the detected endpoint depths are both existent and valid, thereby accurately calculate the current pose while ensuring computational efficiency.

II. RELATED WORK

SLAM originated in 1986 [1], and the development of sensor technology guides to changes in SLAM technology. Combine the sensor with SLAM so that SLAM can be divided into two categories according to the different sensor. One is lidar-based laser SLAM (Lidar SLAM) and the other is visual-based SLAM (Visual SLAM). In recent years, with the continuous enhancement of camera acquisition capabilities, people can acquire a large amount of abundant texture information from the environment, which makes the visual SLAM have stronger scene recognition ability and attract the attention of researchers.

Early visual SLAM programs mostly were filter-based methods, and Extended Kalman Filter (EKF) [7], [8], particle filtering (PF) [9] and Expectation maximization (Expectation maximization), EM) [10], [11] were commonly used. The filter-based SLAM programs have FastSLAM [12], [13] proposed by Montemerlo and MonoSLAM [14] by Daviso2n. FastSLAM used a Rao-Blackwellised filter with the $O(p \log n)$ complexity that p represented the number of particles used and n represented the number of road signs in the map. This method was difficult to accurately control the number of particles to express the pose, too many particles cause a too high time cost, too few particles cause the pose estimation result to be inaccurate. MonoSLAM used EKF to simultaneously estimate camera motion and the three-dimensional structure of unknown environment, and its complexity reached $O(n^3)$. So it was not suitable for running in large scenes. The nonlinear error model of the filtering method and the large computational amount have become the main obstacles for the practical implementation of the filter-based SLAM.

The enormous amount of computation leaded to the emergence of graph-based optimization and maked it be adopted by many SLAM programs. Lu *et al.* proposed the method of graph optimization [15], maintaining the relative spatial relationship between all local data frames and local frames, and expounding the basic model of graph optimization, but it was not implemented due to the calculation conditions at that time. Subsequent researchers discovered that the SLAM problem was actually soluting a sparse matrix, which greatly reduced the computational complexity of graph optimization. The optimization of graphs for SLAM can achieve real-time effects [16]. Strasdat pointed out that the result of graph optimization is better than that of the filter-based method [17]. Bundle Adjustment (BA) was widely used in graph optimization [18]. A fast algorithm for local bundle adjustment for optimization was proposed in [19], which was the first solution to introduce keyframes into monocular vision SLAM and enables the graph optimization method to achieve real-time effects.

Murray and Klein proposed the breakthrough achievement PTAM [20] (Parallel Tracking and Mapping) in the visual SLAM. In this method, pose tracking and mapping were divided into two threads and run in parallel, which was very suitable for small workspaces. The thread separation design of this method has been approved by subsequent studies of SLAM. The most classic one was the ORB-SLAM [2] proposed by Mur-Artal and its extended version ORB-SLAM2 [3], which divided the task into three parallel threads: tracking, local mapping, and closed-loop control. Another PTAM extension was DTSLAM [21], which divided the task into three parallel threads: tracking, mapping, and BA optimization. This method handled pure rotation based on 2D and 3D features and used multiple undefined scale maps without explicit initialization. The method of this type was a classical representation of the feature-based method in visual SLAM, which optimized the pose by minimizing the point reprojection error. However, in the low-texture environment, due to the lack of repetitive and reliable features, it was easy to cause tracking failure or reduce positioning accuracy.

Another method of visual SLAM was the direct method. This kind of method was aimed at the time-consuming

problem of feature extraction and matching in feature method. It was proposed to calculate camera motion only based on image pixel information and used the whole image for alignment. At this time, it can achieve certain effect on parallel processing. Because the direct method was based on the brightness constancy constraint, it was considered that the brightness of the same spatial point on each image was basically invariant during the motion [22]. Newcombe's DTAM (Dense tracking and mapping) [23], one of the representatives of the direct method, can achieve real-time recovery of 3D models, and has high robustness in image blur and feature loss. Engel's LSD-SLAM was different from DTAM, created a semi-dense map on the CPU in real time for pixels with sharp gradients [4]. Engel also proposed the Direct Sparse odometry (DSO) [24], which calibrates the camera exposure time and other parameters to make the method more robust. If the camera hasn't a photometric calibration, the DSO models the camera imaging process and dynamically estimated the photometric parameters during the optimization process. SLAM based on direct method was prone to failure in the environment with poor exposure conditions or fast grayscale changes, and most require GPU to achieve real-time effect. Later, Forster proposed the Semi-direct Visual Odometry (SVO) [25] and the improved and extended version SVO2 [5], which tracked some key-points without calculating key point descriptors, and then estimated camera motion and position based on information around key points like the direct method. The relocation of this method was simple, but the effect is not ideal.

Feature-based methods are faster than the direct method, because feature-based methods used the various features of the image to calculate rather than process the entire image. Some researchers have extended the feature-based visual SLAM, using other features on the image, such as edges, lines, or line segments, to improve the positioning accuracy while ensuring its computational speed. Typically, for example, the line was added into the EKF-SLAM, and the straight-line detection method of this way was to connect the detected feature points for hypotheses and testing to acquire real-time performance [26]. Eade introduced a local block on the edge in the monocular SLAM, and the local block was used to express the edges on the image to avoid tracking problems caused by edge occlusion [27]. Lemaire used line segments in EKF-SLAM, using Plucker coordinates to represent line segments as infinite lines, and line segment extraction was acquired by using a binary image obtained by a gradient filter to connect adjacent high gradient pixel contours and apply line fitting. This method required Plucker constraints to be considered during the update step [28]. Perdices proposed a line-based monocular SLAM solution [29], in which the initialization line segment was derived from singular value decomposition of a matrix composed by a set of planes associated with it. The method can be run in real time only when the number of lines do not exceed 18, which has greater limitations.

Further, researchers combined the point features with line features to express the environment. Lu *et al.* proposed a visual odometer scheme based on point-line RGBD [30]. By analyzing the measurement uncertainty of 3D points and lines, calculate camera motion with the maximum likelihood estimation. Gomez-Ojeda *et al.* proposed a binocular stereo visual odometer scheme [31] based on point-line combination, which used the inverse of point-line features corresponding covariance matrices to weight them, and the covariance was obtained by uncertainty propagation technique of the reprojection function. In this method, the point features were detected by ORB [32] detection and its descriptors, and the line features were detected by LSD [33] and LBD descriptors [34]. Later, the author improved the method and proposed the monocular semi-direct method PL-SVO solution [35]. The SVO was extended to the line feature and detected only when new keyframes were introduced, which improved the real-time performance. After 2017, another tendency in SLAM was utilizing the lidar information. Lidar would capture more accurate results than RGB image, which made the point-line feature extraction even more practical.

For SLAM solutions that utilize line features or point features, there are phenomena of insufficient accuracy or instability in some environments. Therefore, under the guarantee of the original speed of feature method, combining the respective advantages of the point features and line features better, improving the accuracy of the SLAM solution and ensuring the stability of the SLAM are the main objectives of this paper.

III. RPL-SLAM OVERVIEW

The paper proposes an RGBD and Point-Line SLAM (RPL-SLAM) solution based on depth camera and point-line features. The general structure of the RPL-SLAM system is depicted in Figure 1. The process is divided into three main threads, tracking, local mapping and loop closure detection. In the loop detection phase, after the loop correction, an additional global Ba thread is opened to get the optimal structure and motion results of the whole system.

The tracking thread extracts and matches the point and line features of the new frame. Because the depth camera is used, the 3D information of the point and line features can be recovered directly from the collected rgbd image. Therefore, the matching correspondence is 3D-3D correspondence rather than 2D-2D correspondence of the ordinary RGB camera. Use the acquired correspondence to calculate the initial pose and determine whether it is a key frame, and decide when to insert the key frame, and optimize the pose using only motion BA optimization. If it fails, use relocation to recover the new frame. The implementation of relocation requires the location identification thread. The location identification thread is composed of a key frame database that is maintained in real time while the system is running. Here we also add the part of the line. Once the initial pose is acquired and optimized, it will be further optimized using a local visual map derived

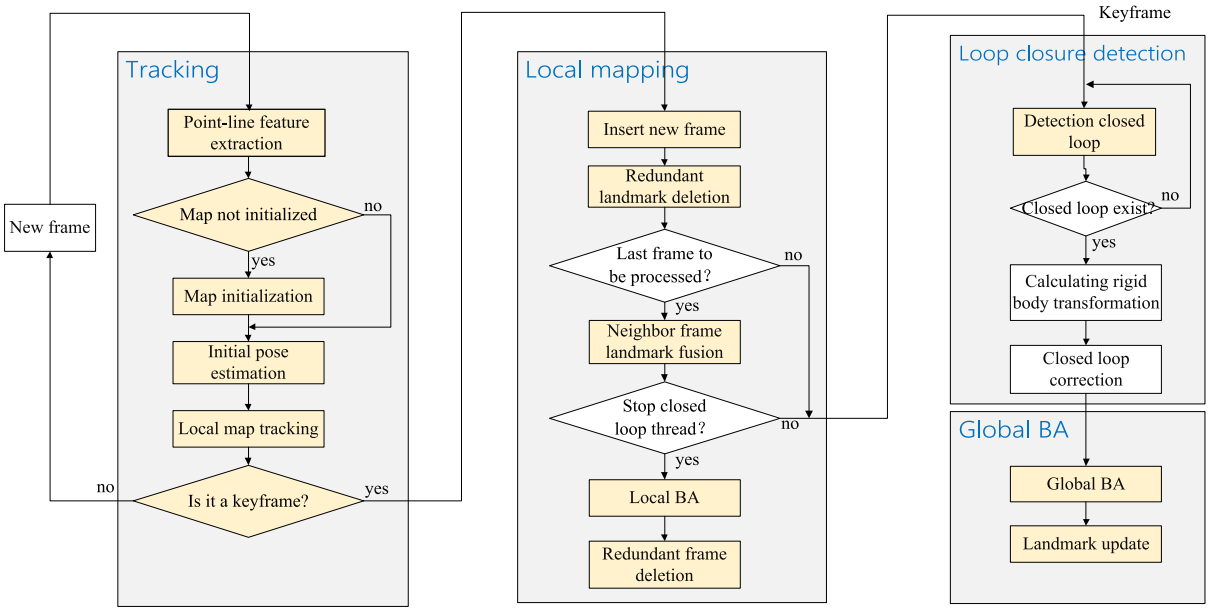


FIGURE 1. RPL-SLAM frame diagram.

from keyframe point-line common view maintained by the system.

The local mapping thread is used to process new keyframes and perform BA optimization to optimize the errors generated during the tracking process. This thread will eliminate redundant roadmaps, maintain endpoints of line segments, and erase redundant keyframes.

Loop closure detection detects the closed loops through the keyframe database maintained by the point-line features, and then reduces the errors that occur during operation through long-running closed-loop constraints, thereby further optimizing the pose. And it is responsible to create new global BA threads to optimize the pose of the key frames on the map to narrow the drift accumulation that occurs during the pose solution to achieve global consistency.

IV. TRACKING OF POINT-LINE FUSION

A. POINT-LINE FEATURE EXTRACTION AND MATCHING

The first step in the tracking thread is to distill the features of the new frame. The adopted mechanism is to extract the point features and line features in parallel. Point features are described by ORB extraction and 256-bit ORB descriptors. For line feature, we propose SLD (Straight Line segment Detector) line segment extraction algorithm. SLD first grays the image, then uses edge detection algorithm to binaries the image. At the detected edge pixel, it connects the line with it's adjacent pixel, and continues to fit the line and extend to the next edge pixel until it meets the collinearity with the current line segment.

In the process of edge detection, we optimize the fixed threshold in the method of [36] and propose an adaptive canny threshold edge detection algorithm. The high and low thresholds are related to the gray distribution of the image. The adaptive Canny edge detection threshold is calculated as shown in formula (1) and formula (2). The threshold is related to the mean value and standard deviation of the pixel gray value of the input gray image. The mean value and standard deviation are calculated as shown in formula (3).

$$Th_{low} = \max\left(1, \lambda * (mean - stdDev)\right) \tag{1}$$

$$Th_{high} = \min\left(254, \lambda * (mean + stdDev)\right) \tag{2}$$

$$mean = \sum_i^N src_i \Big/ N,$$

$$stdDev = \sqrt{\sum_i^N (src_i - mean)^2 \Big/ N}, i = 0, \cdots, N \tag{3}$$

where λ is the scale value, N is the number of pixels in the image. Through large-scale dataset experiments, it is found that the adaptive canny threshold edge detection algorithm can get very good edge detection results in most images whenλ =1.2.

Based on the edge pixel detection, the SLD segment extraction algorithm defines eight possible directions of line chain and the number of adjacent pixels with the edge pixel whose pixel value is not 0 as the seed starting point. After obtaining a linear prediction pixel chain, judge whether the pixels on the current line constitute a straight line. At this time, the straight line segment can be formed by using the straight line prediction pixel chain starting point pixel and the ending point pixel, and the distance from the point in the pixel chain near the straight line to the line is compared with a threshold value to determine whether the pixel point belongs to the line segment. After several line segments are obtained, line segment merging will be generally performed. However, we find that the line segment combination takes a long time. Once the length ratio of the generated line segment is controlled, the

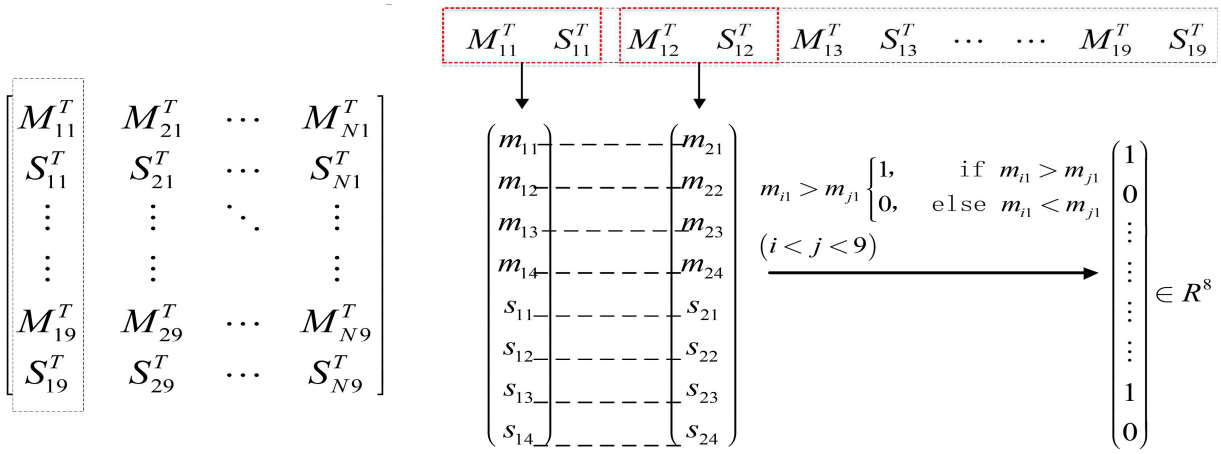


FIGURE 2. Comparison of each group of binary descriptors.

improvement of the line segment combination effect is not obvious. Therefore, the algorithm discards strategy that incremental merging of line segments and limits the number and length of the line segments extracted. Through experiments, it is found that removing the line segments whose pixel sizes are less than the 5% of the minimum of the image size can acquire a good extraction effect, and the extraction speed is greatly improved compared with the classical LSD algorithm.

An improved LBD descriptors is proposed to describe the line segment extracted in this paper. The original LBD descriptors are only 72-dimensional, and the calculation is time-consuming since the original LBD descriptors are not binary and distance between eigenvectors needs to be calculated when the feature is matched. Therefore, in this paper, two elements will be extracted from 32 sets of randomly selected 72-dimensional LBD descriptors in a fixed order and be re-compared to acquire 256-bit vectors consisting of two characters of 0 and 1. For each group, the comparison method is depicted in Figure 2, where N is the number of segments detected in the image. Compare the mean vectors Mj and the standard error vector Sj of two strips which are chosen casually from the nine strips in each 72-dimensional line segment LBD descriptor to acquire an 8-dimensional binary descriptor, compare a total of 32 groups, and then form a 256-dimensional LBD Descriptor. The binary LBD descriptors acquired in the above manner can quickly calculate the Hamming distance between two eigenvectors by using Binary XOR and other operations, thereby accelerating the process of line segment matching.

Similar to point feature matching using bag-of-words to find corresponding relationship [37], we propose a line segment matching DBM method based on bag-of-words (LBD Bow Matcher, DBM). The initial correspondence between line features is acquired by descriptor comparison between those features belonging to the same node at level 4 in the vocabulary tree (counting up from the leaves). The matching scheme limits the matching of features on the same node, which greatly reduce the matching range and increase

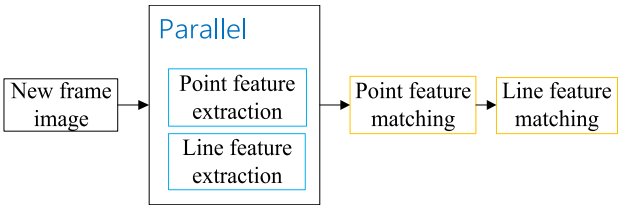


FIGURE 3. Point line extraction and matching process.

the matching efficiency. In the matching, mismatches are inevitable. In the same way that the ORB considers the geometric verification to eliminate the mismatch, the condition that linear rotation constraint to vote with the range of the angle variation between the two matching lines is adopted, and then the straight linear matching relationship can be eliminated that does not belong to the highest range of three angle variations.

In the process, the entire point-line features extraction is divided and matched into two stages, and a series-parallel processing mode is adopted. Firstly, the parallel thread is used to extract the point features and line features respectively. After that, since the cost of context switching between threads is greater than the time of using point-line matching, the serial strategy is adopted in the second stage that point-line feature matching. The schematic diagram of the process is depicted in Figure 3.

B. INITIAL POSE ESTIMATION

After the map is initialized, if the previous frame is successfully tracked, the current frame acquires the initial pose estimation from the previous frame by the motion model. Then find out if the number of interior points supporting the pose is sufficient. For the line segment, after using the line segment matching method proposed above, the corresponding matching line will be contacted with the spatial line.

If the motion model is invalid, use the map points and lines of the adjacent reference keyframe to track the current frame pose to check if there are enough correspondences to support

that pose is correct. If there are enough correspondences, it is assumed that the initial value of the current frame pose is the same as the pose of the previous frame tracked successfully, and use motion-only BA to optimize partially, and then find out whether the number of interior points supporting the pose is sufficient. If it is not enough, the method of judging the current frame pose by the adjacent frame fails, and the tracking is lost. If the tracking is lost, convert the frame to a point-line dictionary and inquire the keyframe database for relocation candidates.

Considering that the line segments play a major role in the textureless conditions, but the points generally play a major role in the texture-rich area, calculating the optimal relocation score for an image demands to formulate corresponding point-line weighted scores, as depicted in equation (4).

$$Score_{pl} = \left(n_p s_p + n_l s_l\right) \Big/ \left(n_p + n_l\right) \tag{4}$$

where s_p, s_l are the similarity scores between the two images calculated using the bag-of-words, and n_p, n_l are the number of points and lines, respectively.

The pose is solved by the point-line correspondence between relocation candidates and then optimized by using motion-only BA.

Motion-only BA optimizes camera rotation matrix and displacement. Minimize the reprojection error and line reprojection error between the matched 3D points and feature points in the world coordinate system, as depicted in equation (5).

$$\{R, t\} = \arg\min_{R,t} \left(\sum_{i \in \mathcal{X}} \rho_s \left\| x^i - \pi_s \left(R X^i + t \right) \right\|^2_{\Sigma_s} + \sum_{j \in P} \rho_l \left\| e_l \left(l^j, \pi_l \left(\mathcal{HL}^j \right) \right) \right\|^2_{\Sigma_l} \right) \tag{5}$$

where R and t are the poses that need to be optimized currently. x^i, l^j are the feature points and lines detected of the current image.

π_s, π_l are projection functions of 3D points and lines projected onto the current image, respectively. ρ_s, ρ_l are the robust Huber cost functions of points and lines, respectively. $\sum_s, \sum_l$ is the corresponding covariance matrix of the points and lines, respectively.

e_l is the reprojection error between l^j and the line which is between the projection $\pi_l \left(\mathcal{HL}^j \right)$.

C. LINE SEGMENT REPROJECTION ERRORS REPRESENTED BY PLUCKER

1) PLUCKER REPRESENTATION OF THE SPATIAL LINE

The three-dimensional spatial line cannot be represented by the four-dimensional vector of the homogeneous coordinates. And the line can be defined as the connection of two points or the intersection of two planes so that there are four freedom degrees. So the lines need to be represented by five-dimensional vectors. However, the five-dimensional linear vector cannot be linked to the points and lines represented by four-dimensional vectors with mathematical formulas. To overcome this problem, Plucker is applied to represent the spatial line and the orthogonal representation of the line for the back-end optimization, and update the orthogonal representation of the line with a minimum of four parameters.

Due to the usage of the depth camera, the depth of the line segment endpoint can be read from the depth map, and the detected line segment endpoints will be initialized only if their depth exists and is valid. Assume that the spatial homogeneous coordinates of the detected line segment endpoints are $A \sim \left(\tilde{A}\ m\right)$ and $B \sim \left(\tilde{B}\ n\right)$, where $\sim$ represents the homogeneous equivalence of the projective space, then the Plucker coordinate $\mathcal{L}^T \sim \left(n^T\ v^T\right) \in R^6$ of the line is depicted in equation (6).

$$\begin{cases} n = \tilde{A} \times \tilde{B} \\ v = m\tilde{B} - n\tilde{A} \end{cases} \tag{6}$$

where, $\tilde{A}, \tilde{B}, n, v \in R^3$, $\times$ represents the cross product of two vectors. The distance between the line and the origin is d, and d satisfies $d = \|OQ\| = \|n\| \big/ \|v\|$.

2) LINE SEGMENT REPROJECTION ERRORS BASED ON PLUCKER

When using Plucker to represent the spatial line, the line reprojection error model and the predicted line that the line projects to the image are depicted in Figure 4. As can be seen in Figure 4, the center O_c of the camera and the spatial line form the plane π and the plane intersects the image plane to obtain a projection line l_c of Plucker. It is easy to see that the straight lines on the plane π are all projected onto l_c, so the projection line is irrelevant to the component v of the spatial line Plucker coordinates and is related to the component n of that. If the camera internal parameter K is known, project the Plucker line to the image plane as depicted in equation (7).

$$l_c = \tilde{K} \cdot n_c = cof\ (K) \cdot n_c = \begin{bmatrix} f_x & 0 & 0 \\ 0 & f_y & 0 \\ -f_x c_x & -f_y c_y & f_x f_y \end{bmatrix} \cdot n_c \tag{7}$$

where, $\tilde{K}$ is the Plucker projection matrix and cof represents the cofactor matrix.

Therefore, the reprojection error of the lines is as depicted in the formula (8). In Figure 4, the error is defined as the distance from the endpoints P_s, P_e of the observed line to the projection line l_c. Since the observation is four-dimensional, the last 2 dimensions of the error are zero when implementing the algorithm.

$$\begin{aligned} \mathbf{e}_l &= \left[e_1\ e_2 \right]^T \\ &= \left[\left(l^T \cdot P_s\right) \Big/ \sqrt{l_1^2 + l_2^2}\ \left(l^T \cdot P_e\right) \Big/ \sqrt{l_1^2 + l_2^2} \right]^T \end{aligned} \tag{8}$$

D. LOCAL MAP TRACKING

After getting the initial pose estimation, we can associate the map with the current frame. If using a large global map, projecting the corresponding 3D points and lines to the current frame is not applicable to real-time SLAM because of the

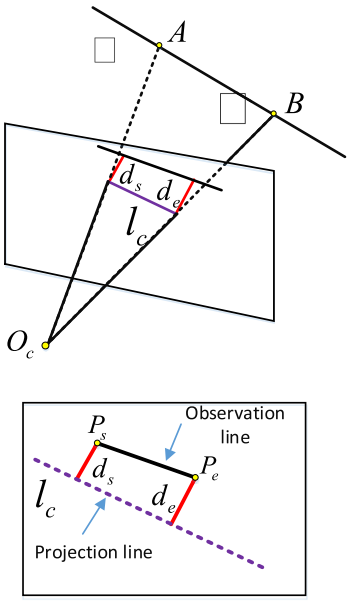


FIGURE 4. Plucker linear projection and projection error model representation.

high computational complexity and computational cost. This paper uses the method of maintaining local maps to projection pursuit. The maintained local map consists of three parts: the keyframe collection $\{K_1\}$ with the same 3D points and 3D lines as the current frame, adjacent keyframe collection $\{K_2\}$ in the point-line common view, reference frames $\{K_{ref}\}$ which contains the most map points and lines with the current frame and belongs to the keyframe collection$\{K_1\}$. Search the map points and lines which are matched with the current frame from the local map, and these points and lines do not contain map points and lines that have been confirmed to be observed by the current frame. These map points and lines in the local map are projected onto the image plane of the current frame according to the current frame pose estimation.

There may be occlusion in the line segment, for which it is difficult to determine whether the map line is in the current image. Therefore, discuss the relationship between the two endpoints of the line segment and the camera position in three cases. In the camera coordinate system, the two 3D endpoints may be in front of the camera, either behind the camera or in case that one is in the front and the other is in the back. If they are both behind the camera, they cannot be projected to the current frame. If they are both in front, they must be projected into the image of the current frame.

If it is a tandem situation, it is assumed that L_{front} is the endpoint in front of the camera plane and L_{back} is the endpoint behind the camera plane. The line connecting of two endpoints intersect with the normalized plane z=1 in front of the camera to form an intersection point. It is considered that the projection of the intersection point onto the image plane is the projection line segment endpoint of the line segment which consists L_{front} and L_{back}. The projection point of L_{front} on the image is the other endpoint of the projection line segment. However, the projection point of L_{front} and the previous point of intersection may be considered to be the same point, which means that the spatial line is perpendicular to the image plane. At this time, the projection of the line to the image plane is only one point, indicating that the map line is not observed by current frame. Considering that the intersection point $L_{intersection}$ that the line whose one of endpoints is in front of the camera and the other one is behind the camera intersects with the normalized plane z=1 can be depicted in equation (9), the line can be transformed to the line in camera normalization plane, and the situation is converted to the situation that the two endpoints are both in front of the camera.

$$L_{intersection} = L_{front} + \mu \left(L_{front} - L_{back} \right), \mu \in (0, 1) \quad (9)$$

Projecting the endpoint onto the image plane needs to considerate that the endpoint of the projection line may completely exceed the extent of the image plane. At this point, crop them using linear clipping algorithm.

V. BACKEND OPTIMIZATION BASED ON THE POINTS AND LINES

A. MAP MANAGEMENT

There may be mismatching problems in feature matching, leading to the mistaken triangulation of landmarks (points, lines), and such landmarks will increase the computation time of backend BA optimization. Therefore, eliminating bad landmarks and limiting the size and dimensions of optimized landmarks can improve the accuracy and efficiency of BA optimization. The elimination principle mainly includes two points. Firstly, the number of frames that track the landmark must be greater than 25% of the predicted visible frames, otherwise, it is eliminated. Secondly, the landmark passes no less than 3 keyframes from the creation to the tracking thread, but the number of keyframes observing it is less than 3, then it is eliminated.

Besides, the same landmark will be observed repeatedly during the continuous tracking process. These different observations of the same landmark by adjacent keyframes associated with the common view at the second level may generate duplicate map points or lines or no corresponding map points associated. So, further check and merge these map points or lines are needed.

For each map line that matches the current frame, there are two results of finding line features in the adjacent key frames associated with these common views at the second level. One is that no line features are found, the other is that the line features that have been associated with other map lines are considered to be the observation of current processing map lines in adjacent keyframes. For the first case, associate the current processing map line with the line feature. For the second case, merge the map line associated with the line feature with the current processing map line, and then select the map line with the most observations as the current updated map line.

Further determine whether the map line can be projected to the adjacent keyframe to be processed. If it can, the

line segment matching of the line features of the adjacent keyframe is carried out to find the best matching. For each line feature that may be considered to be the best match, the current map line needs to be projected to the current processing adjacent keyframe using Plucker coordinates, and then the line projection error will be used to determine whether the error square sum of the distance from the endpoint of the current line feature to the projected line is greater than the threshold (this paper is set to 5.991). Only the line features whose error square sum is smaller than the threshold may be considered to be the projection of the current map line in current processing adjacent keyframe. Then select the line features that are considered to be the best match of the map line, and check whether there is already a map line associated with them. If there is already an associated map line, the line segment is merged, otherwise the line feature is associated with the map line.

B. BA OPTIMIZATION

Local BA is to optimize the map model which is composed of a part of poses and landmarks both selected from the map, optimizing both the pose and the location of the landmark. There are many strategies for keyframes selected by local BA optimization. The keyframes selected for optimization in this paper are the keyframe collection $\{K_L\}$ adjacent to the current processing keyframes in the point-line common view and the collection $\{P_L, L_L\}$ of landmark observed by these keyframes. The keyframes also include the keyframe collection K_F that can observe a part of the landmarks of the collection $\{P_L, L_L\}$ but is not adjacent to the current frame in the common view. During the optimizing process, the pose of K_F is not optimized and is only used to optimize the position of the landmark in space within the cost function. The formula for local BA optimization is depicted in (10).

$$\{X_i, Y_i, R_l, t_l\} = \underset{X_i, Y_i, R_l, t_l}{\arg\min} \sum_{k \in K_L \cup K_F} \left(\sum_{j \in X_k} \rho\left(E_{pkj}\right) + \sum_{h \in Y_k} \rho\left(E_{lkh}\right) \right) \tag{10}$$

where X_k and Y_k are the 3D point match set and the 3D line match set of the landmark collections $\{P_L\}$ and $\{L_L\}$ in the keyframe k, respectively. E_{pkj} and E_{lkh} represents reprojection error of points and lines, respectively.

Corresponding to the local BA, the global BA, a special case of the local BA, is to optimize all the keyframes together with all the landmarks of the map. The initial keyframe is fixed by the pose, which is not optimized during the optimization process.

C. LINE SEGMENT INFORMATION UPDATE

After the BA optimization optimizing the pose and the landmark position, it is needed to update the participating keyframes and landmarks. In the method of this paper, the line segment information needs to be updated accordingly due to the addition of line segment features. Since only a portion of the spatial line segment is observed during the tracking process, the endpoints of the line segment also need to be maintained constantly. The basic principle of line segment endpoint maintenance is depicted in Figure 5.

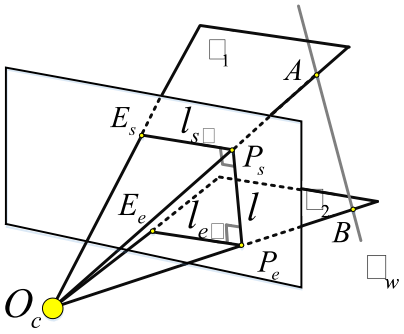


FIGURE 5. Spatial straight line segment maintenance.

Assuming that there is a straight line L in the space, the line L detected in the image is considered to be an observation of the spatial straight line projection onto the image. Then, on the image plane, use the endpoints P_s and P_e of the line segment to make the vertical lines $l_{s\perp}, l_{e\perp}$ for the detection line segment l respectively. So, the endpoints A and B of the spatial line segment are the intersections of the spatial straight line $\mathcal{L}_w$ and the planes π_1, π_2 which are respectively formed by P_s, P_e, the points E_s, E_e on the vertical lines $l_{s\perp}, l_{e\perp}$ and the camera center O_c, respectively. The distance $d\left(E_s, P_s\right), d\left(E_e, P_e\right)$ of the two endpoints of the vertical lines $l_{s\perp}, l_{e\perp}$ can be any value. The plane π is calculated by the four-dimensional homogeneous coordinates X_1, X_2, X_3 of three arbitrarily corresponding spatial points on the plane, as depicted in equation (11), where the superscript $\sim$ indicates that the last dimension of the homogeneous coordinates is removed. Calculate the straight line Plucker matrix $\mathbf{L}$ using the straight line $\mathcal{L}_w$ according to the formula (12), and then substitute it and the formula (11) into the formula (13) to find the intersection point as the endpoint of the straight line.

$$\pi = \begin{bmatrix} \left(\tilde{X}_1 - \tilde{X}_2\right) \times \left(\tilde{X}_2 - \tilde{X}_3\right) \\ -\tilde{X}_3^T \left(\tilde{X}_1 \times \tilde{X}_2\right) \end{bmatrix} \tag{11}$$

$$\boldsymbol{L} \sim \begin{pmatrix} [n]_\times & -v \\ v^T & 0 \end{pmatrix} \tag{12}$$

$$D = \boldsymbol{L}\pi \tag{13}$$

Not only after the BA optimization, but also when adding an observation of a keyframe to the map line, and after the pose graph is optimized for the corresponding pose, the line segment needs to be updated, that is, maintaining the line endpoint. The distance between the two endpoints to be updated will be updated only when it is greater than the currently saved distance.

D. LOOP CLOSURE DETECTION

Due to the presence of noise, the error produced by estimating the pose of each frame inevitably passes and accumulates to subsequent frames. With long-term operation, this error

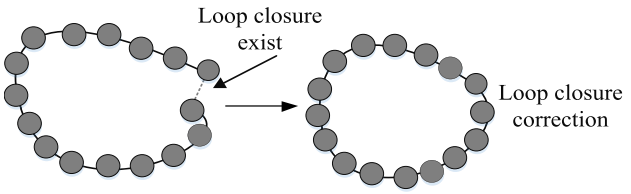


FIGURE 6. Pose graph optimization example.

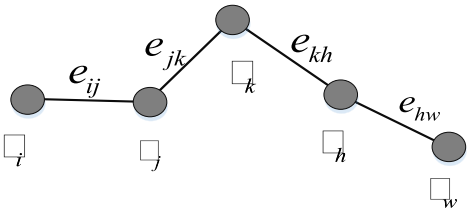


FIGURE 7. Pose graph optimization example.

accumulation will become larger, and it is hard to guarantee that the consistency of long-running SLAM. Loop closure detection can detect that a scene has been reached so that there is a constraint between long-term frames (frames from the start frame to the loopback frame), which can reduce the previously accumulated error.

When running the loop closure detection, first take out a keyframe from the maintained keyframe queue to be processed, and the keyframe queue is composed of keyframes passed in the partial mapping. If the keyframe queue is not empty, select a keyframe from queue for processing to detect whether it is a loop closure frame. Otherwise, detect whether the loop closure process should be terminated. If the process should not be terminated, wait for a certain time to continue to determine whether the keyframe queue is empty.

If there is a loop closure, the current frame will be matched with the loop closure candidate to obtain a point-line correspondence. In a texture-rich environment, calculate the rigid body transformation by using the correspondence of points preferentially, otherwise using the line-based correspondence. Whether it is based on the correspondence of points or the correspondence of lines, at least three sets of correspondences are needed to solve.

After calculating the pose transformation of two frames, the pose graph optimization (PGO) can be performed, and the pose graph optimization is depicted in Figure 6. The node of the pose graph is the pose of the keyframe, and the edge is the estimated error of the relative motion between the two pose nodes as depicted in Figure 7. The error is defined as depicted in equation (14).

$$e_{ij} = \ln\left(\exp\left(\left(-\xi_{ij}\right)^{\wedge}\right)\exp\left(\left(-\xi_i\right)^{\wedge}\right)\exp\left(\xi_j^{\wedge}\right)\right)^{\vee} \quad (14)$$

When the candidate between the current frame and the loop closure frame is calculated, correct the loop closure using the pose graph and apply the corrected result to the participating keyframes as well as the map points and lines associated with the keyframes. Then another thread is opened immediately to perform the global BA optimization in order to reduce the cumulative error.

TABLE 1. Time-consuming comparision of different line segment extraction algorithms (unit: ms).

Method of extraction / Picture number	LSD	EDLine	SLD（Ours）
1	42.16	9.05	**7.80**
2	67.18	15.72	**13.04**
3	46.71	8.77	**6.79**
4	25.08	8.11	**5.71**

VI. RESULTS AND ANALYSIS

This paper applies the TUM RGB-D benchmark [38] to compare the current mainstream visual SLAM method with the RPL-SLAM system proposed in this paper. The experimental environment is Intel Core I7-7700HQ (8 cores @2.8 GHz) with 16 GB of memory.

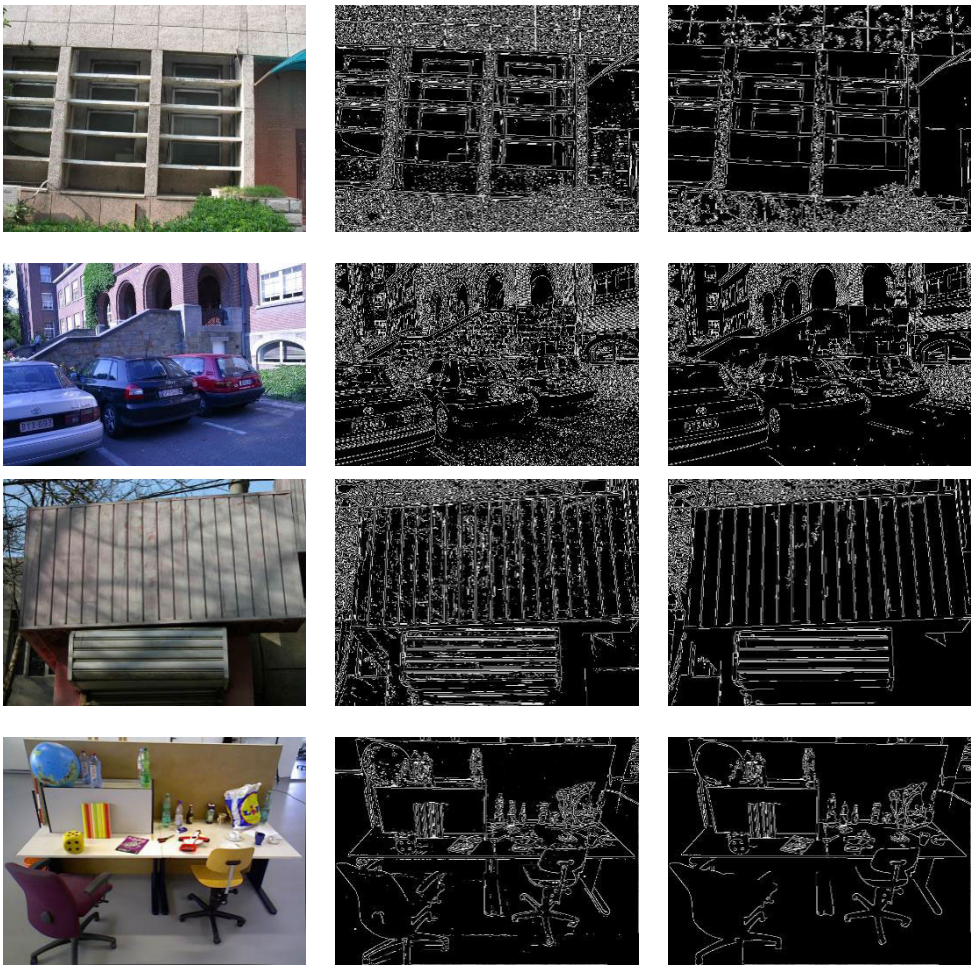
A. POINT-LINE MATCHING

First, for the edge detection algorithm, the adaptive threshold Canny are compared with the fixed threshold method Canny (Both high and low are 50). The experiment uses four sets of data, three of which are from building rotation, lenven, occlusion in [39] and another from a set of images in the TUM dataset. The edge detection result is depicted in Figure 8. It can be observed that the Canny edge detection algorithm which adaptively calculates the high and low thresholds can eliminate more weak edges and is suitable for most scenes, so it lays a foundation for eliminating noise and improving efficiency both in the subsequent line segment extraction.

Compare the line segment extraction algorithm LSD [33] or EDLine [40] with the SLD method proposed in this paper and define that line segments larger than 40 pixels is meaningful. By extracting straight lines in the same image, it can be found that the meaningful line segment ratio obtained by the SLD method is significantly higher than that of LSD and EDLine, as depicted in Figure 9.

Besides, the algorithm efficiency of the SLD line segment extraction algorithm and LSD and EDLine algorithms are compared and analyzed, and the four groups of images are tested separately. The test results are depicted in Table 1. From Table 1, we can see that the extraction speed of SLD is significantly faster than the other two line segment extraction algorithms.

For the comparison of line segment matching methods, the line segment matching method based on bag-of-words DBM proposed in this paper are compared with the brute



(a) Original image (b) Original Canny (c) Adaptive Canny

FIGURE 8. Comparison of adaptive Canny algorithm and ordinary Canny algorithm.

(a) LSD extraction（18.3%） (b)EDLine extraction（49.4%） (c)SLD extraction（70.4%）(Ours)

FIGURE 9. Results of different line extraction algorithms and the proportion of meaningful line segments.

force matching method BFM with cross-validation and the binary descriptor matching method BDM based on MIH (Multi-Index Hashing) [41]. The matching result is depicted in Figure 10.

As shown in Table 2, we compare and analyze the total matching line segment number (TMN), the correct rate (CR, the correct matching quantity/total matching quantity) and the matching total consumption time (TC) of the above three methods. The matching number should be as much as possible. However, real-time should be considered more in visual SLAMs that require real-time performance, so the total consumption time should be as short as possible.

TABLE 2. Comparison of four line segment matching experiments.

	TMN			CR			TC（ms）		
	BFM	BDM	DBM (Ours)	BFM	BDM	DBM (Ours)	BFM	BDM	DBM (Ours)
Case 1	65	**67**	45	0.97	0.95	**1**	0.31	2.38	**0.13**
Case 2	**55**	**55**	31	0.98	0.98	**1**	0.50	3.36	**0.15**
Case 3	16	**25**	15	0.63	0.56	**0.73**	0.22	1.99	**0.09**
Case 4	**78**	**78**	56	**1**	**1**	**1**	0.18	1.14	**0.07**

（a）BFM

（b）BDM

（c）DBM(Ours)

FIGURE 10. Comparison of three matching methods for the building_rotationd picture group.

Mismatching will affect the accuracy of positioning, so the accuracy higher, the better.

It can be seen from the experiment that the based on the bag-of-words line segment matching algorithm proposed by this subject is better than the other three algorithms, and the matching accuracy and efficiency is higher than other algorithms, which is critical for real-time SLAM. The line segment matching method based on bag-of-words can guarantee the real-time requirement of SLAM, and the method based on the linear rotation constraint can eliminate the mismatching.

Because this paper uses both the point features and the line features and uses the series-parallel cross processing mode, the serial and parallel time for the same image point-line extraction and matching are tested and compared. The results

TABLE 3. Efficiency comparison of ponit-line extraction and matching in serial or parallel(ms).

Squence	Serial	Parallel	Ours
1	36.92	28.86	**28.27**
2	52.94	36.65	**36.31**
3	44.22	31.04	**30.73**
4	27.15	17.51	**17.31**

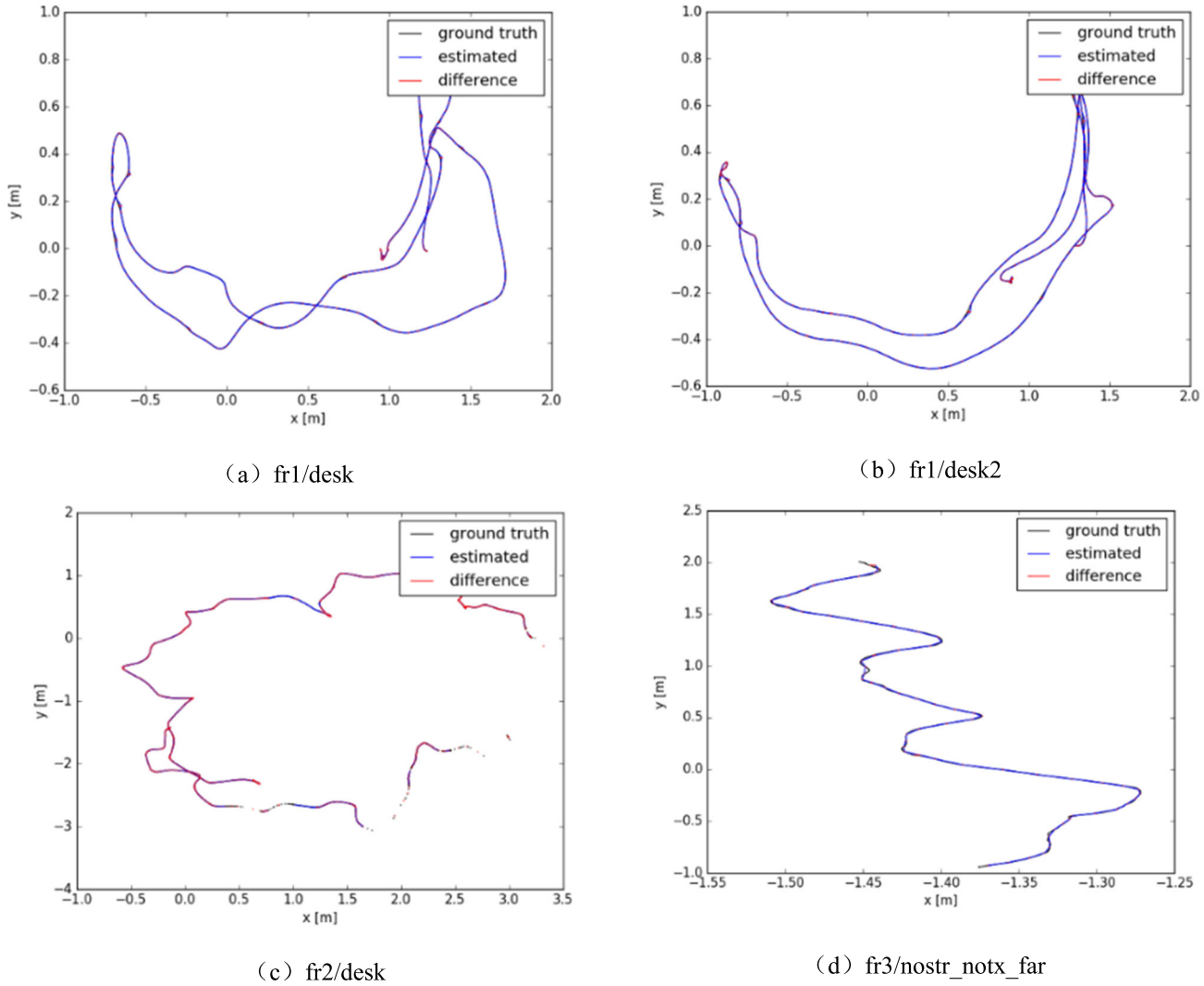
are depicted in Table 3. The execution efficiency of the three cases is compared where the extraction and matching are serial, all the parallel and the parallel extraction before serial matching method proposed in this paper.

According to the result of the comparison, the method proposed in this paper is more efficient. This is because in the texture-rich case, the extraction scale of points and lines is large and the calculation time is long, parallel extraction can save time. However, the time for the point-line matching is short, which makes it difficult to ignore the thread switching time, so parallel cannot improve the efficiency of the point-line matching, and serial can ensure sufficient efficiency.

B. POSITIONING RESULTS

In order to verify the positioning accuracy of the proposed method, the TUM benchmark data set is applied to compare RPL-SLAM method proposed in the paper with the RGBD version of ORB-SLAM2. Figure 11 is a comparison of the pose error of the partial dataset RPL-SLAM in TUM with the real trajectory.

The TUM RGBD benchmark data set provides two evaluation methods for SLAM to evaluate the positioning accuracy, which are relative pose error (RPE) and absolute trajectory error (ATE). The RPE estimates the difference between motion and real motion to evaluate the visual odometer or drift error with a loop closure SLAM system. ATE is used to align the estimated trajectory with the real trajectory, and then directly evaluates the pose error, which is suitable for evaluating the pose estimation accuracy of the visual SLAM system. Table 4 shows the data comparison between



（a）fr1/desk　（b）fr1/desk2

（c）fr2/desk　（d）fr3/nostr_notx_far

FIGURE 11. Comparison of the estimates of the partial TUM dataset RPL-SLAM with the real trajectory, which is a comparison of fr1/desk, fr1/desk2, fr2/desk, fr3/nostr_notx_far sequences.

TABLE 4. TUM RGBD positioning accuracy benchmark ATE and RPE.

TUM RGBD Sequences	ORB-SLAM2		RPL-SLAM（Ours）	
	ATE	RPE	ATE	RPE
fr1/desk	0.017	0.018	**0.009**	**0.012**
fr1/desk2	0.021	0.033	**0.012**	**0.021**
fr1/room	0.041	0.038	0.048	**0.033**
fr1/floor	0.026	0.013	0.039	0.045
fr2/desk	0.008	0.006	**0.007**	**0.005**
fr2/xyz	0.004	0.0016	0.010	**0.012**
fr3/nostr_notx_far	0.018	0.019	**0.008**	**0.007**
fr3/long_office	0.010	0.0073	0.012	**0.0063**
fr3/str_tx_near	0.012	0.014	0.014	0.023

ATE and RPE of RPL-SLAM method and RGBD version of ORB-SLAM2.

In order to further verify the effect of RPL-SLAM method, we compared it with RGBD version of ORB-SLAM2 on ICL-NUIM dataset, as shown in Table 5.

TABLE 5. ATE and RPE of ICL-NUIM dataset.

ICL-NUIM Sequences	ORB-SLAM2		RPL-SLAM（Ours）	
	ATE	RPE	ATE	RPE
lr kt0	0.012	0.0081	**0.011**	0.0090
lr kt1	0.101	0.0075	**0.008**	**0.0045**
lr kt2	0.014	0.0090	0.018	**0.0051**
of kt0	0.023	0.0123	0.025	**0.0098**
of kt1	0.057	0.0290	**0.023**	**0.0104**
of kt2	0.010	0.0065	0.026	0.0085
of kt3	0.129	0.0044	**0.012**	**0.0038**

It can be seen from Table 4 and Table 5, the RPL-SLAM method proposed in this paper is superior to the RGBD version of ORB-SLAM2 on the vast majority of sequences in the two data sets of TUM RGBD and ICL-NUIM. However, in a few datasets, the positioning accuracy of the RPL-SLAM method has decreased. The reason is that in some images with

rich texture information, there is inevitably a false alarm in the straight line extraction, which leads to an increase in system noise and an increase in error when solving, so that reduce the positioning accuracy. In response to this problem, we will further look for ways to optimize in subsequent research centers.

C. REAL TIME ANALYSIS

For slam technology, positioning accuracy is an important index to measure its quality, and at the same time, we hope to maintain a high calculation efficiency. In the previous point line matching experiment, through the test and analysis of multiple data sets, we can find that our proposed line feature extraction algorithm has high efficiency, which lays the foundation for the real-time performance of SLAM algorithm.

Using the standard data set TUM RGBD, we further test and analyze the tracking time in our RPL-SLAM method, and compare it with the ORB-SLAM2 scheme. Through experiments, the average tracking time of ORB-SLAM2 is 35.9ms, and the average tracking time of RPL-SLAM is 39.7ms. Since only point features exist in ORB-SLAM2, and we add line features based on point features, the overall tracking time is increased. Although time increases, the increase is within the predictable range, and the matching accuracy is effectively improved. The average tracking time of RPL-SLAM is 39.7ms and the processing frame rate reaches 25fps with the addition of line features. The whole algorithm improves the positioning accuracy and still meets the real-time requirements.

VII. CONCLUSION

In this paper, a new scheme RPL-SLAM is proposed, which uses depth images and point-line features to improve the accuracy and reliability of camera trajectory estimation. The line segment extraction algorithm SLD and the line segment matching algorithm DBM are proposed, which improve the accuracy of point-line matching while ensuring real-time performance. In addition, the Plucker coordinates expressing infinite length is applied to express the spatial line segment and use the orthogonal representation to update the spatial line and optimize the back-end, for which the back-end optimization error model of point-line fusion can be unified to solve the problem that instability in optimization. RPL-SLAM is a complete visual SLAM solution which combines depth images and point-line features. The effectiveness of the proposed solution is verified on TUM benchmark dataset. For future research, further integration of inertial measurement unit information can be considered to improve the robustness of dynamic environment and pure rotation.

REFERENCES

[1] H. Durrant-Whyte and T. Bailey, "Simultaneous localization and mapping: Part I," *IEEE Robot. Autom. Mag.*, vol. 13, no. 2, pp. 99–110, Jun. 2006.

[2] R. Mur-Artal and J. D. Tardós, "ORB-SLAM: Tracking and mapping recognizable features," in *Proc. Workshop Multi View Geometry Robot.*, CA, USA, Jul. 2014, pp. 1–6.

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

[4] J. Engel, T. Schöps, and D. Cremers, "LSD-SLAM: Large-scale direct monocular SLAM," in *Proc. Eur. Conf. Comput. Vis.*, Zürich, Switzerland, 2014, pp. 834–849.

[5] C. Forster, Z. Zhang, M. Gassner, M. Werlberger, and D. Scaramuzza, "SVO: Semidirect visual odometry for monocular and multicamera systems," *IEEE Trans. Robot.*, vol. 33, no. 2, pp. 249–265, Apr. 2017.

[6] J. Fuentes-Pacheco, J. Ruiz-Ascencio, and J. M. Rendón-Mancha, "Visual simultaneous localization and mapping: A survey," *Artif. Intell. Rev.*, vol. 43, no. 1, pp. 55–81, Jan. 2015.

[7] P. Moutarlier and R. Chatila, "An experimental system for incremental environment modelling by an autonomous mobile robot," in *Experimental Robotics I*. Berlin, Germany: Springer, 1990, pp. 327–346.

[8] P. Moutarlier and R. Chatila, "Stochastic multisensory data fusion for mobile robot location and environment modelling," in *Proc. Int. Symp. Robot. Res.*, Tokyo, Japan, May 1989, p. 207.

[9] S. Thrun, D. Fox, W. Burgard, and F. Dellaert, "Robust Monte Carlo localization for mobile robots," *Artif. Intell.*, vol. 128, nos. 1–2, pp. 99–141, May 2001.

[10] H. Shatkay and L. P. Kaelbling, "Learning topological maps with weak local odometric information," in *Proc. 15th Int. Joint Conf. Artif. Intell. (IJCAI)*, Nagoya, Japan, 1997, pp. 920–927.

[11] S. Thrun, W. Burgard, and D. Fox, "A probabilistic approach to concurrent mapping and localization for mobile robots," *Mach. Learn.*, vol. 31, no. 1, pp. 29–53, 1998.

[12] M. Montemerlo, S. Thrun, D. Koller, and B. Wegbreit, "FastSLAM: A factored solution to the simultaneous localization and mapping problem," in *Proc. AAAI Nat. Conf. Artif. Intell.*, AB, Canada, 2002, pp. 593–598.

[13] M. Montemerlo, S. Thrun, D. Koller, and B. Wegbreit, "FastSLAM 2.0: An improved particle filtering algorithm for simultaneous localization and mapping that provably converges," in *Proc. Int. Joint Conf. Artif. Intell. (IJCAI)*, Acapulco, Mexico, 2003, pp. 1151–1156.

[14] A. J. Davison, I. D. Reid, N. D. Molton, and O. Stasse, "MonoSLAM: Real-time single camera SLAM," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 29, no. 6, pp. 1052–1067, Jun. 2007.

[15] F. Lu and E. Milios, "Globally consistent range scan alignment for environment mapping," *Auto. Robots*, vol. 4, no. 4, pp. 333–349, 1997.

[16] E. Mouragnon, M. Lhuillier, M. Dhome, F. Dekeyser, and P. Sayd, "Monocular vision based SLAM for mobile robots," in *Proc. 18th Int. Conf. Pattern Recognit. (ICPR)*, Hong Kong, 2006, pp. 001–006.

[17] H. Strasdat, J. M. M. Montiel, and A. J. Davison, "Visual SLAM: Why filter?" *Image Vis. Comput.*, vol. 30, no. 2, pp. 65–77, Feb. 2012.

[18] B. Triggs, P. F. McLauchlan, R. I. Hartley, and A. W. Fitzgibbon, "Bundle adjustment—A modern synthesis," in *Proc. Int. Workshop Vis. Algorithms*. Berlin, Germany: Springer, 1999, pp. 298–372.

[19] E. Mouragnon, M. Lhuillier, M. Dhome, F. Dekeyser, and P. Sayd, "Real time localization and 3D reconstruction," in *Proc. IEEE Comput. Soc. Conf. Comput. Vis. Pattern Recognit. (CVPR)*, New York, NY, USA, Jun. 2006, pp. 363–370.

[20] G. Klein and D. Murray, "Parallel tracking and mapping for small AR workspaces," in *Proc. 6th IEEE ACM Int. Symp. Mixed Augmented Reality*, Nara, Japan, Nov. 2007, pp. 1–10.

[21] D. C. Herrera, K. Kim, J. Kannala, K. Pulli, and J. Heikkila, "DT-SLAM: Deferred triangulation for robust SLAM," in *Proc. 2nd Int. Conf. 3D Vis.*, vol. 1, Dec. 2014, pp. 609–616.

[22] T. Bill and Z. Andrew, "Vision algorithms: Theory and practice," in *Proc. Int. Workshop Vis. Algorithms*, Corfu, Greece, Sep. 1999, pp. 298–375.

[23] R. A. Newcombe, S. J. Lovegrove, and A. J. Davison, "DTAM: Dense tracking and mapping in real-time," in *Proc. Int. Conf. Comput. Vis.*, Barcelona, Spain, Nov. 2011, pp. 2320–2327.

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

[25] C. Forster, M. Pizzoli, and D. Scaramuzza, "SVO: Fast semi-direct monocular visual odometry," in *Proc. IEEE Int. Conf. Robot. Autom. (ICRA)*, Hong Kong, May 2014, pp. 15–22.

[26] P. Smith, I. Reid, and A. J. Davison, "Real-time monocular SLAM with straight lines," in *Proc. Brit. Mach. Vis. Conf.*, Edinburgh, U.K., 2006, pp. 17–26.

[27] E. Eade and T. Drummond, "Edge landmarks in monocular SLAM," *Image Vis. Comput.*, vol. 27, no. 5, pp. 588–596, Apr. 2009.

[28] T. Lemaire and S. Lacroix, "Monocular-vision based SLAM using line segments," in *Proc. IEEE Int. Conf. Robot. Autom.*, Roma, Italy, Apr. 2007, pp. 2791–2796.
[29] E. Perdices, L. M. López, and J. M. Cañas, "LineSLAM: Visual real time localization using lines and UKF," in *Proc. 1st Iberian Robot. Conf.* Cham, Switzerland: Springer, 2014, pp. 663–678.
[30] Y. Lu and D. Song, "Robust RGB-D odometry using point and line features," in *Proc. IEEE Int. Conf. Comput. Vis. (ICCV)*, Santiago, Chile, Dec. 2015, pp. 3934–3942.
[31] R. Gomez-Ojeda and J. Gonzalez-Jimenez, "Robust stereo visual odometry through a probabilistic combination of points and line segments," in *Proc. IEEE Int. Conf. Robot. Autom. (ICRA)*, Stockholm, Sweden, May 2016, pp. 2521–2526.
[32] E. Rublee, V. Rabaud, K. Konolige, and G. Bradski, "ORB: An efficient alternative to SIFT or SURF," in *Proc. Int. Conf. Comput. Vis.*, Nov. 2011, vol. 11, no. 1, p. 2.
[33] R. G. Von Gioi, J. Jakubowicz, J.-M. Morel, and G. Randall, "On straight line segment detection," *J. Math. Imag. Vis.*, vol. 32, no. 3, pp. 313–347, Nov. 2008.
[34] L. Zhang and R. Koch, "An efficient and robust line segment matching approach based on LBD descriptor and pairwise geometric consistency," *J. Vis. Commun. Image Represent.*, vol. 24, no. 7, pp. 794–805, Oct. 2013.
[35] R. Gomez-Ojeda, J. Briales, and J. Gonzalez-Jimenez, "PL-SVO: Semi-direct monocular visual odometry by combining points and line segments," in *Proc. IEEE/RSJ Int. Conf. Intell. Robots Syst. (IROS)*, Daejeon, South Korea, Oct. 2016, pp. 4211–4216.
[36] J. H. Lee, S. Lee, G. Zhang, J. Lim, W. K. Chung, and I. H. Suh, "Outdoor place recognition in urban environments using straight lines," in *Proc. IEEE Int. Conf. Robot. Autom. (ICRA)*, Hong Kong, May 2014, pp. 5550–5557.
[37] R. Mur-Artal and J. D. Tardos, "Fast relocalisation and loop closing in keyframe-based SLAM," in *Proc. IEEE Int. Conf. Robot. Autom. (ICRA)*, Hong Kong, May 2014, pp. 846–853.
[38] J. Sturm, N. Engelhard, F. Endres, W. Burgard, and D. Cremers, "A benchmark for the evaluation of RGB-D SLAM systems," in *Proc. IEEE/RSJ Int. Conf. Intell. Robots Syst.*, Vilamoura, Algarve, Oct. 2012, pp. 573–580.
[39] K. Li, J. Yao, M. Lu, Y. Heng, T. Wu, and Y. Li, "Line segment matching: A benchmark," in *Proc. IEEE Winter Conf. Appl. Comput. Vis. (WACV)*, Lake Placid, NY, USA, Mar. 2016, pp. 1–9.
[40] C. Akinlar and C. Topal, "EDLines: A real-time line segment detector with a false detection control," *Pattern Recognit. Lett.*, vol. 32, no. 13, pp. 1633–1642, Oct. 2011.
[41] M. Norouzi, A. Punjani, and D. J. Fleet, "Fast exact search in Hamming space with multi-index hashing," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 36, no. 6, pp. 1107–1119, Jun. 2014.

DAN LI received the B.E. and M.S. degrees in mechanical design, manufacturing and automation and the Ph.D. degree in computer science from the Huazhong University of Science and Technology (HUST), Wuhan, China, in 1998, 2002, and 2008, respectively. She is currently an Associate Professor with the School of Computer Science and Technology, HUST. Her researches span computer graphics, multimedia, and intelligence.

SHUANG LIU received the bachelor's degree from the Huazhong University of Science and Technology, Wuhan, China, in 2018, where he is currently pursuing the master's degree in computer science and technology with the School of Computer Science and Technology. His researches span computer graphics and computer vision.

WEILAI XIANG received the bachelor's degree from the Northwestern Poly-Technique University of Science and Technology in 2020. He is currently pursuing the master's degree in computer science and technology with the School of Computer Science and Technology, Huazhong University of Science and Technology, Wuhan, China. His researches span computer graphics and virtual reality.

QIWEI TAN received the bachelor's degree from the College of Software, Central South University, Changsha, China, in 2017. He is currently pursuing the master's degree in computer science and technology with the School of Computer Science and Technology, Huazhong University of Science and Technology, Wuhan, China. His research interest includes computer vision.

KAICHENG YUAN is currently pursuing the bachelor's degree in computer science and technology with the School of Computer Science and Technology, Huazhong University of Science and Technology, Wuhan, China. His researches span computer graphics and computer vision.

ZHEN ZHANG is pursuing the bachelor's degree in computer science and technology with the School of Computer Science and Technology, Huazhong University of Science and Technology, Wuhan, China. Her research spans computer graphics.

YINGSONG HU received the Ph.D. degree in computer science and technology from the School of Computer Science and Technology, Huazhong University of Science and Technology (HUST), Wuhan, China, in 2011. He is currently an Associate Professor with the School of Computer Science and Technology, HUST. His researches span image processing and computer vision.