A Robust RGB-D SLAM System With Points and Lines for Low Texture Indoor Environments

Qiang Fu, Hongshan Yu, Lihai Lai, Jingwen Wang, Xia Peng, Wei Sun, and Mingui Sun, *Member, IEEE*

***Abstract*— High-performance 6D pose estimation and dense 3D mapping with an RGB-D camera has recently attracted substantial research attention since this type of camera can simultaneously capture RGB and depth information. However, there is an unresolved problem in estimating pose and generating highly accurate 3D maps from challenging indoor scenes. This paper presents a real-time simultaneous localization and mapping (SLAM) system based on the RGB-D camera for indoor mobile robots. Our contributions are fourfold. First, we propose a complete high-accuracy SLAM system based on a combination of information from points and lines, which differs from most solutions that rely on only point features. Second, we propose a novel pose solver to handle point and line correspondences, in which a line-based inliers refinement (LBIR) algorithm is proposed to remove outliers. Third, we construct a unified optimization model to concurrently minimize point and line reprojection errors, and extend it to the bundle adjustment (BA) method. Fourth, extensive experiments demonstrate the robustness, accuracy, and real-time performance of the proposed system on public TUM datasets and real world scenes. The empirical results show that the proposed system achieves a comparable or better performance than state-of-the-art methods. Notably, our system can operate in nearly texture-less scenes, while other methods are prone to failure.**

***Index Terms*— SLAM, RGB-D camera, challenging indoor scenes, point and line features, mobile robots, 3D mapping.**

I. Introduction

ONE of the goals of robotics is a mobile robot that can operate autonomously in the real world. To meet this goal, a location and map are the most important prerequisites [1], [2]. The visual Simultaneous Localization and Mapping (SLAM) technique is an attractive solution because it does not require a priori information with respect to the surroundings [3], [4].

Despite the considerable progress in this field of research [4]–[8], there is still an unresolved problem in estimating pose and generating highly accurate 3D maps from indoor scenes. Indoor environments can be quite challenging due to: 1) Variable illumination; 2) low texture. To overcome these challenges, vision-based SLAM methods have received substantial research attention in the last two decades. Since Parallel Tracking and Mapping (PTAM) [5] was introduced in 2007, many real-time solutions have been proposed including optical flow-based methods, DVO [6], LSD-SLAM [7], and so on, and feature-based methods, such as RGB-D SLAM-v2 [9], ORB SLAM2 [10], and so on. Note that these systems are primarily built using point features, such as the well-known SIFT [43], SURF [44], and Oriented FAST and Rotated BRIEF (ORB) [23]. These point extractors do not perform as well in low-texture scenes, which means the systems cannot always maintain efficiency. On the other hand, based on the structural regularity of indoor environments, Line Segment Detector (LSD) can be used to extract corresponding line features. However, computing pose only using line correspondences is less reliable than using point correspondence, because line features are more sensitive to partial occlusions [11], [12].

A comparison of representative SURF, ORB, and LSD [24] algorithms is shown in Fig. 1, value of the decision threshold in ORB is set to the standardized value of 20 and its number threshold value is set to 1000. The other extractors follow predefined patterns. From this figure, the point features extracted using SURF and ORB extract are poor, and prone to be lost to track. The results reveal that the methods using a combination of point and line features might yield better accuracy and robustness than methods that only use point features.

In this paper, we exploit the application of line features on the basis of point features for a SLAM system with an RGB-D camera. We also introduce several works aimed at further increasing the robustness and accuracy of current RGB-D SLAM systems.

Our contributions are fourfold:

- A novel and complete RGB-D SLAM system using points and lines is proposed. This system is different from most solutions that only use points.
- A unified optimization model is constructed to concurrently minimize point and line reprojection errors, and it is further extended to the bundle adjustment (BA).

Manuscript received June 24, 2019; revised June 29, 2019; accepted July 2, 2019. Date of publication July 8, 2019; date of current version October 4, 2019. This work was supported in part by the National Natural Science Foundation of China under Grant 61573135 and Grant U1813205, in part by the National Key Technology Support Program under Grant 2015BAF11B01, in part by the Key Research and Development Project of Science and Technology Plan of Hunan Province under Grant 2018GK2021, in part by the Hunan Key Laboratory of Intelligent Robot Technology in Electronic Manufacturing under Grant 2018001, in part by the Science and Technology Plan Project of Shenzhen City under Grant JCYJ20170306141557198, in part by the Key Project of Science, in part by the Key Project of Science and Technology Plan of Changsha City under Grant kq1801003, in part by the Hunan Natural Science Foundation under Grant 2017JJ3118, in part by the Aviation Science Foundation Projects under Grant 201705W1001, and in part by the National Institutes of Health of the United States under Grant R01CA165255 and Grant R21CA172864. The associate editor coordinating the review of this paper and approving it for publication was Dr. You Li. *(Corresponding author: Hongshan Yu.)*

Q. Fu, H. Yu, L. Lai, J. Wang, and W. Sun are with the National Engineering Laboratory for Robot Visual Perception and Control, Hunan University, Changsha 410082, China, and also with the Shenzhen Institute of Hunan University, Shenzhen 518057, China (e-mail: cn.fq@qq.com; yuhongshancn@163.com; 1356332601@qq.com; wjw0375@hnu.edu.cn; david-sun@126.com).

X. Peng is with the School of Information and Science, Central South University, Changsha 410083, China (e-mail: 931278889@qq.com).

M. Sun is with the Laboratory for Computational Neuroscience, University of Pittsburgh, Pittsburgh, PA 15260 USA (e-mail: drsun@pitt.edu).

Digital Object Identifier 10.1109/JSEN.2019.2927405

TABLE I
THE STATISTIC OF POPULAR VISUAL SLAM SYSTEMS

Systems	Camera Types	Direct Methods	Indirect Methods		Pose Estimation	Relocation	Dense Map
			Point Feature	Line Feature			
Engel *et al.* [7]	Mono	√	×	×	√	×	×
Forster *et al.* [18]	Mono	√	×	×	√	×	×
Forster *et al.* [17]	Multicamera	√	×	×	√	×	×
Newcombe *et al.* [19]	RGBD	√	×	×	√	×	√
Whelan *et al.** [4]	RGBD	×	×	×	×	×	√
Klein *et al.* [5]	Mono	×	√	×	√	×	×
Davison *et al.* [21]	Mono	×	√	×	√	×	×
Labbé *et al.* [8]	RGBD	×	√	×	√	√	√
Endres *et al.* [9]	RGBD	×	√	×	√	√	√
Mur-Artal *et al.* [10]	RGBD/Stereo/Mono	×	√	×	√	√	×
Pumarola *et al.* [11]	Mono	×	√	√	√	√	×
Gomez-Ojeda *et al.* [12]	Stereo	×	√	√	√	√	×
Ours	RGBD	×	√	√	√	√	√

Elastic Fusion* builds 3D dense environment based on point cloud information. Relocation* can be implemented by loop closure module theoretically.

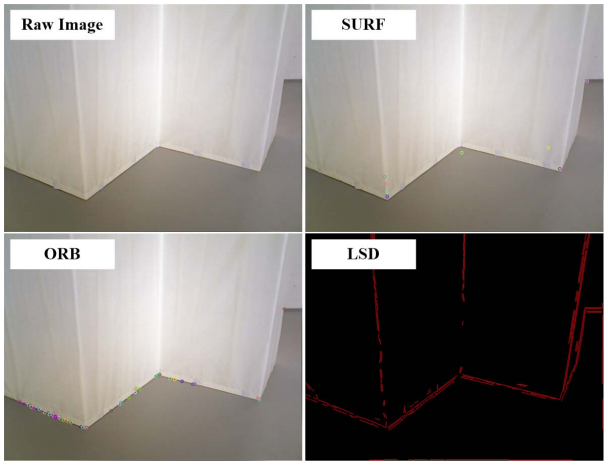

Fig. 1. Popular point and line feature extractors in challenging indoor low-texture scenes. The well-known SURF, ORB, and LSD algorithms were compared, in which decision threshold in ORB is set to standardized value of 20 and the number threshold value is set to 1000. According to this figure, SURF and ORB extract poor point features, which are prone to lost tracking. Therefore, a methods than combines point and line features might yield a better accuracy and robustness than method only using point features.

- A novel pose solver is proposed to obtain an accurate relative transformation matrix from matching pairs of points and lines. In the solver, which is aimed at line outlier problem, we present the Line-Based Inliers Refinement (LBIR) algorithm based on the PnL method and the PnP method.
- Comprehensive experiments are carried out on public TUM datasets and real scenes to demonstrate the robustness, accuracy, and real-time of the proposed system. The quantitative evaluations corroborate that our system achieves better performance than state-of-the-art methods in most of the sequences. Notably, our system can operate in extremely challenging scenes while other methods are prone to failure.

The remainder of the paper is organized as follows: Section II details related work. Section III presents the system architecture. Section IV introduces point and line-based SLAM methods. Sections V, VI, and VII describe the three threads of the proposed system: *tracking*, *local mapping*, and *loop closure and mapping,* respectively. The Experiments are detailed in Section VIII. Finally, concluding remarks and the highlights of future works are provided in Section IX.

II. RELATED WORK

Pose estimation and 3D mapping in unknown environments has been an extremely important research area in robotics for decades [13], [14]. The Visual SLAM approach provides a solution by recovering the camera trajectory from an image sequence, which has many applications in autonomous robot navigation and augmented reality (AR) [15], [16].

Currently, visual SLAM can be divided into two classes: photometric-based direct methods and feature-based indirect methods [17]. Table I provides statistics of current popular visual SLAM systems in terms of camera type, direct or indirect method, whether the system provides functions of pose estimation or relocation, and whether a dense 3D map is output.

A. Photometric-Based Direct Methods

Direct methods are built on a strong assumption of invariant illumination, and pose is directly estimated by minimizing the image pixel-level intensity error [18]. Early methods were computationally intensive, which was typically addressed using GPU parallelism, because they implement whole image alignment, such as LSD SLAM [7] and DTAM [19]. To speed up the process, a semi-dense approach was proposed in [20], in which the depth is only estimated for pixels with high intensity gradients. Based on sparse optimization, SVO-v1 [18] and SVO-v2 [17] were proposed quick estimation with little weight and few burdens. However, direct methods have a common disadvantage in eliminating accumulative error, compared with indirect methods.

B. Feature-Based Indirect Methods

Indirect methods estimate a camera pose and build a structure of environment based on the assumption of invariant features in the image. Therefore, robustness, accuracy and

efficiency of a system depend on the performance of a selected feature [8]–[10]. Based on the camera type, indirect methods can be divided into Mono SLAM, stereo SLAM, and RGBD SLAM. The three systems are developed using a similar same scheme, composed of three main threads: *tracking*, *local mapping*, and *loop closure*. In short, the *tracking* thread tracks image features to solve an initial pose, the *local mapping* thread usually optimizes the pose through minimizing local error, the *loop closure* thread provides a correct, global, and powerful data association for system.

The largest difference in the three types of systems is how depth is estimated. Mono SLAM cannot estimate an absolute scale unless an extra sensor [21], [29], such as Inertial Measurement Unit (IMU). Stereo SLAM utilizes stereo parallax relations to estimate depth [2], this method requires at least two cameras. Small distance between pair of cameras will result in poor estimation accuracy for far-away objects. RGBD SLAM have emerged which capture both RGB image and depth information simultaneously [9]. Thus, this type of camera provides a powerful solution to solve the scale problem.

The accuracy and robustness of an indirect methods is known to depend on the performance of the feature extractor [5], [8], i.e., poor feature extracting in images will directly lead to a failure [14]. In this paper, indirect methods are divided into point-based SLAM, line-based SLAM, and fusion-based SLAM according to the type of feature extractor.

1) Point-Based SLAM: Point-based SLAM methods are the current mainstream methods, such as PTAM [5], RGBD SLAM-v2 [9], ORB SLAM2 [10], and so on. In 2007, when Klein and Murray proposed the first point-based SLAM system using two parallel threads, tracking and mapping, for real-time application [5], point-based SLAM began to be a research focus [23]. To boost the accuracy of pose estimation, Mur-Artal *et al.* proposed the improved ORB algorithm [23] to obtain a uniform distribution of points [10]. Endres *et al.* utilized point correspondences and RGB-D camera properties to build a dense 3D map [9]. Although considerable number of solutions have been proposed, point-based SLAM solutions still has difficulties in maintaining high performance in low-texture scenes. The performance of the point feature is a key for SLAM solutions that use points.

2) Line-Based SLAM: In line-based SLAM frames, the LSD algorithm is commonly used as a line extractor [24], and a 3D line is typically represented by segments with two end points. Therefore, it is applicable in small workspace environments in which the endpoints do not disappear from the camera view [25]–[28]. However, computing pose only from line correspondences is less reliable than from point correspondence [29]. In addition, it is difficult for these systems that only use lines to find a correct loop closure since line feature is not unique enough for scene recognition [12]. Therefore, these methods hardly solve the problem of relocation when a robot loses its position, and accumulative global error produced by long run can affect the performance of the system [30]. Point-based SLAM system can handle these problems using the loop closure module [8]–[10].

3) Fusion-Based SLAM: Fusion of point and line features has recently been utilized in the SLAM community recently. Which has shown, that line features can be used to improve the performance of a point-based SLAM system. Pumarola *et al.* proposed a real-time mono SLAM system with points and lines [11], which built upon the ORB SLAM2. However, it is difficult for this system to operate in low-texture scenes due to poor point feature extraction. Ruben *et al.* proposed a stereo SLAM system using points and lines [12], which was developed for application in low-textured environments. The largest contribution of the system is proposing a point-line based loop closure module. Although the novel loop closure method increases robustness, the higher computation cost makes this system difficult to reach real-time. In [31], Lee *et al.* developed a low-cost embedded system for indoor service robots based on vanishing point and line landmarks. However, this system does not provide 6D pose estimation. In addition, a common disadvantage of all of these methods is that a dense map module is not provided.

Inspired by the previous works, RGBD SLAM using points and lines is proposed in this paper. The system can output 6D pose estimation and a 3D map in some challenging indoor sequences, while other solutions are prone to failure.

III. System Overview

In this paper, we focus on high-performance robot 6D pose estimation and 3D mapping in challenging indoor scenes. The proposed system differs from most solutions that only use point features by estimating pose and building a 3D map based on a combination of point and line correspondences. Specifically, we use an improved ORB algorithm as the point extractor, which is based on our previous work [14], and LSD is used as the line extractor.

The general structure of the proposed system is depicted in Fig. 2. The pipeline is based on many state-of-the-art solutions (e.g., PTAM [5], ORB SLAM [10]), and the main modules consist of three threads: *tracking, local mapping,* and *loop closure and mapping*.

1) *Tracking:* This thread estimates the current robot pose between the current and previous frames. First, point and line features from the current frame are extracted and registered with the features in the previous frame. Subsequently, according to the associations, this thread quickly computes an initial relative motion matrix T_r^c and more reliable inliers using the proposed pose solver. Finally, the current robot pose can be obtained by computing $T_{cw} = T_r^c T_{rw}$, where T_{cw} and T_{rw} represent the current camera pose and previous (reference) camera pose, respectively. The details are described in Section V.
2) *Local Mapping:* This thread minimizes the local errors using BA method, when a keyframe is inserted into the system. This action is only performed in keyframes to reduce computation, similar to many SLAM systems (e.g., RGBD SLAM-v2 [9]). The details are described in Section VI.
3) *Loop Closure and Mapping:* In this thread, we reduce the accumulated errors during exploration through loop detection and loop correction. Thus our system can build

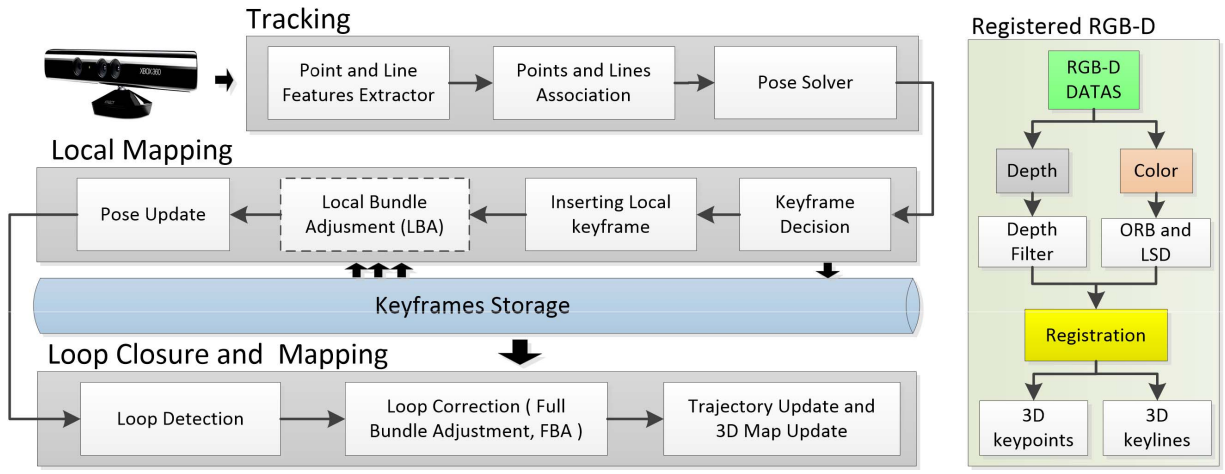

Fig. 2. System overview. Our system is composed of three main threads: *tracking, local mapping,* and *loop closure and mapping*. The *tracking* thread quickly estimates a robot pose using the information of point and line associations. The *local mapping* thread determines whether to add a new keyframe and optimizes it with local BA. The *loop closure and mapping* thread is constantly checking for loops and correcting them with full BA. The keyframe storage provides previous image information for *local mapping* and *loop closure and mapping*.

a 3D map satisfying the constraint of global consistency. The details are described in Section VII.

IV. Point and Line-Based SLAM Methods

In our RGB-D SLAM system that uses points and lines, the line-based solution creates new problems, such as reliable line matching and pose estimation. For line matching problem, we present the LBIR algorithm, by exploring the similarity between the PnP problem and PnL problem; Subsequently, for pose estimation, we construct a standardized optimization model to concurrently minimize point and line reprojection errors.

A. PnP vs PnL

1) PnP Problem: Let a 3D reference point be $P_i^w = [x_i^w, y_i^w, z_i^w]^T$ in world frame, which is denoted as $P_i^c = [x_i^c, y_i^c, z_i^c]^T$ in camera frame. The correspondence 2D projection of the point on image plane is $\mu_i = [u_i, v_i]^T$. Using pinhole camera model, we have:

$$\begin{bmatrix} s_i u_i \\ s_i v_i \\ s_i \end{bmatrix} = K_c \begin{bmatrix} x_i^c \\ y_i^c \\ z_i^c \end{bmatrix} = K_c T_w^c \begin{bmatrix} x_i^w \\ y_i^w \\ z_i^w \end{bmatrix} \tag{1}$$

where K_c represents the 3×3 Camera Intrinsic Matrix, which can be obtained in advance by camera calibration; s_i is the scaling factor, which can be directly determined by the depth measurements in this work. $T_w^c = \left[R_w^c | t\right]$ is the relative motion matrix, also called Camera Extrinsics, and consists of a 3×3 rotation matrix R_w^c and a 3×1 translation vector t that aligns the camera and world coordinate frames,. There is also the 3D point $P_i^c = R_w^c P_i^w + t$. s_i can be eliminated by substituting the third row of (1) into the first row and the second row. Therefore, each point pair $P_i^c \longleftrightarrow \mu_i$ has two constraints equation.

Following the EPnP algorithm [32], P_i^w can be rewritten in terms of the barycentric coordinates of four control points C_j^w, $j = 1, 2, 3, 4$, selected to define an orthonormal basis centered at the origin of the world coordinate system. Therefore, every reference point can be expressed as $P_i^w = \sum_{j=1}^4 \alpha_{ij} C_j^w$. Note that the barycentric coordinates α_{ij} are independent in the coordinate system, and specifically remain the same when writing the reference points in the camera coordinate system. Therefore, we have

$$P_i^c = \sum_{j=1}^4 \alpha_{ij} C_j^c \tag{2}$$

Combining (1), (2), and $K_c = \begin{bmatrix} f_u & 0 & u_c \\ 0 & f_v & v_c \\ 0 & 0 & 1 \end{bmatrix}$, we have

$$\begin{bmatrix} s_i u_i \\ s_i v_i \\ s_i \end{bmatrix} = \begin{bmatrix} f_u & 0 & u_c \\ 0 & f_v & v_c \\ 0 & 0 & 1 \end{bmatrix} \sum_{j=1}^4 \alpha_{ij} \begin{bmatrix} x_j^c \\ y_j^c \\ z_j^c \end{bmatrix} \tag{3}$$

Subsequently, s_i is eliminated by substituting the third row of (1) into the first row and the second row, thus, equation (3) yields two linear equations for each point:

$$\begin{cases} \sum_{j=1}^4 \alpha_{ij} f_u x_j^c + \alpha_{ij}(u_c - u_i) z_j^c = 0 \\ \sum_{j=1}^4 \alpha_{ij} f_v y_j^c + \alpha_{ij}(v_c - v_i) z_j^c = 0 \end{cases} \tag{4}$$

Next, let $x = \left[C_1^{cT}, C_2^{cT}, C_3^{cT}, C_4^{cT}\right]^T$ be an unknown 12×1 vector consisting of the control point coordinates in the camera coordinate system, and concatenate constraint equations of n point correspondences. A linear system is given by

$$Mx = 0 \tag{5}$$

where M is a $2n \times 12$ matrix (each point correspondence has two constraint equations), generated by arranging the coefficients of (4) for n points.

2) PnL Linear Formulation: Following [33], let (L_i^w, S_i^w) be the endpoints of a 3D reference line in the world frame and (L_i^c, S_i^c) in the camera frame. Camera observation of the line can be expressed as a projection plane passing through the

camera center and the line. Let n_i be the normal of the corresponding projection plane. Then, following (2), we express the two endpoints of a line by using the barycentric coordinates

$$\begin{cases} L_i^c = \sum_{j=1}^{4} \alpha_{ij} C_j^c \\ S_i^c = \sum_{j=1}^{4} \beta_{ij} C_j^c \end{cases} \tag{6}$$

where α_{ij} *and* β_{ij} are the barycentric coordinates which are similarly independent in the coordinate system. By concatenating the coordinates of 3D control points $\{C_j^c$, an unknown 12×1 vector $x = [C_1^T, C_2^T, C_3^T, C_4^T]^T$ can be obtained, which is the same as the process in the PnP problem. As a result, we obtain the same formulation as (5), the PnP linear formulation. Note that in this work we transform a 3D line for two 3D endpoints, and each point is expressed using the barycentric coordinates, where the barycentric coordinates are independent in the camera and world coordinate system.

3) Line-Based Inliers Refinement Method: We obtain the same formulation as the PnP linear formulation (5) by transforming a 3D line for two 3D endpoints. Inspired by the PnP work of [34], in this work we present an inliers refinement scheme for the PnL problem. Camera pose can be iteratively optimized by minimizing the following objective function:

$$x = \arg\min_x ||W(x)Mx||^2 \quad s.t.\ ||x||^2 = 1 \tag{7}$$

where $W(x) = \begin{bmatrix} w_1 & \cdots & 0 \\ \vdots & \ddots & \vdots \\ 0 & \cdots & w_{2n} \end{bmatrix}$, which indicates that $W(x) = diag(\ldots, w_i, \ldots)$ is a $2n \times 2n$ diagonal matrix where $w_i = 0 or 1$, and its diagonal elements depend on x. $w_i = 0$ means the correspondence is an outlier and $w_i = 1$ means the correspondence is an inlier. Initially, we assume all correspondences are inliers. Therefore, $W(x)$ is initialized to as an identity matrix.

Following (5), let $\tau = Mx$ be the residual vector, and for i_{th} line correspondence, $\epsilon_i = ||[\tau_{2i-1}, \tau_{2i}]||$ is associated with the corresponding algebraic error. Thus, all the elements of matrix $W(x)$ are updated by

$$w_i = \begin{cases} 1 & \text{if } \epsilon_i < \varepsilon_\tau \\ 0 & \text{otherwise,} \end{cases} \tag{8}$$

where ε_τ is a threshold used to determine whether a correspondence is an inlier or an outlier.

Inspired by the work of [28] and [34], ε_τ can be given by

$$\varepsilon_\tau = \rho Q_{K\%}(\epsilon_1, \epsilon_2, \ldots, \epsilon_n), \tag{9}$$

where ρ denotes a number coefficient, and $Q_{K\%}(\epsilon_1, \epsilon_2, \ldots, \epsilon_n)$ returns the algebraic error of the correspondence that is at the boundary of the lowest K percent. This function is used as a robust estimator to reject correspondences with the largest algebraic errors. In the experiment of [34] for point correspondences, $\rho = 1$ and $K = 25$ yield satisfying robustness. With respect to line correspondences, we found that $\rho = 2$ *and* $K = 25$ yield satisfying robustness. x and $W(x)$ are updated iteratively until convergence of the target function $||W(x)Mx||^2$. Empirically, the algorithm usually converges in less than five iterations.

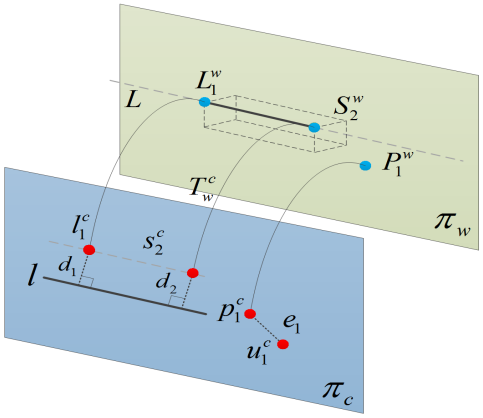

Fig. 3. Illustration of reprojection errors with lines and points.

B. Point and Line-Based Optimization Model

1) Line Based Reprojection Error Function: Let $L_i^w and S_i^w$ be 3D endpoints of the i_{th} line in the world frame π_w, and $l_i and s_i$ be their 2D detections in the image frame π_c. As shown in Fig. 3, assume $L_1^w \in L_i^w$ and $S_1^w \in S_i^w$ are the 3D endpoints of line L in π_w and l is the 2D detection line in π_c corresponding to L. Following (1), we have $l_1^c = K_c T_w^c L_1^w$, $s_1^c = K_c T_w^c S_1^w$, where l_1^c and s_1^c are 2D reprojection points in π_c.Thus, the error e_1 between the 2D l_1^c and the 2D l can be expressed by the distance $d_1(p, l)$ of point-to-line, we have

$$e_1 = d_1\left(l_1^c, l\right) = \lambda^T K_c T_w^c L_1^w \tag{10}$$

where λ is the normalized line coefficients with respect to point-to-line. In [33], λ can be given by

$$\lambda = \frac{l_d^h \times s_d^h}{\left|l_d^h \times s_d^h\right|} \tag{11}$$

where $(l_d, s_d) \in R^2$ are the 2D endpoints of the 2D detection line l, and $(l_d^h, s_d^h) \in R^3$ are their corresponding homogeneous coordinates.

For n lines, the line reprojection error is defined as the sum of point-to-line distances d_i between the projected line segment endpoints and the detected line in the image frame. e_L can be computed by

$$e_L = \sum_1^n d(l_i^c, l) + \sum_1^n d\left(s_i^c, l\right). \tag{12}$$

Note that, in practice, the error form of L_i^w is the same as S_i^w. Therefore, by substituting S_i^w with L_i^w, equation (12) can be reduced to

$$e_L = \sum_1^{2n} d\left(l_i^c, l\right) = \frac{1}{2} \sum_{i=1}^{2n} ||\lambda^T K_c T_w^c L_i^w||^2 \tag{13}$$

2) Point and Line-Based Reprojcetion Error Target Function: For m points, the 3D-2D reprojection error e_P with respect to point correspondences can be given by

$$e_P = \frac{1}{2} \sum_{j=1}^{\mathrm{m}} ||u_j^c - \frac{1}{s_j} K_c T_w^c P_j^w||^2 \tag{14}$$

where P_j^w are the 3D points in π_w, and u_j^c is the detection 2D points corresponding to P_j^w, s_j is the projection scale factor.

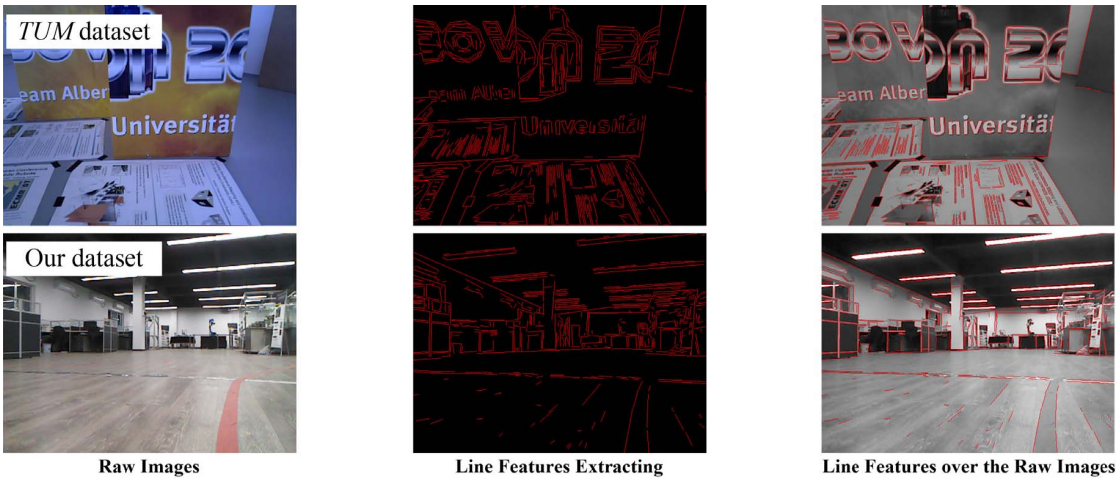

Fig. 4. Illustration of LSD algorithm in *TUM* and our own dataset.

As a result, assuming there are n lines and m points and combining (13) and (14). The target function of the optimization model can be given by

$$e = \frac{1}{2}\sum_{i=1}^{2n} ||\lambda^T K_c T_w^c L_i^w||^2 + \frac{1}{2}\sum_{j=1}^{m} ||u_j^c - \frac{1}{s_j} K_c T_w^c P_j^w||^2. \quad (15)$$

With the target function, iterative algorithms can be applied in SLAM to solve the optimization problem, which could be implemented using a general optimization framework g2o [38]. In this paper, we adopt the Gaussian-Newton algorithm to compute the descent gradient of the target function, and solve the optimization problem by the g2o framework. In addition, we extend the point and line-based optimization model to bundle adjustment (BA), the details are introduced in Section VI-C.

V. Tracking

In this section, we introduce the *tracking* thread. Based on the inlier refinement algorithm and optimization model in section IV, we propose a novel pose solver to determine an accurate pose estimation from point and line correspondences. The details of the solver are introduced in Section V-C.

A. Point and Line Feature Extractor

Our point feature extractor is based on ORB [23], which are oriented multiscale FAST corners with an associated 256-bit descriptor that have a good invariance to viewpoint, therefore, the algorithm allows the system to match them with wide baselines.

To ensure a homogeneous distribution, we follow the work in [10] by dividing each scale level into a grid, and attempt to extract at least five corners per cell. Then, we detect corners in each cell. Note that our system adopts the Adaptive Threshold FAST (AT-FAST) method to extract corners in challenging scenes (e.g., low or high texture) efficiently. AT-FAST is an improved FAST method from our previous work [14], where the threshold ϵ_{FAST} can be given by

$$\epsilon_{FAST} = k * (\frac{1}{m}\sum_{i=1}^{m} I_{i_{max}} - \frac{1}{m}\sum_{i=1}^{m} I_{i_{min}})/I_{i_{avg}} \quad (16)$$

where $I_{i_{max}}$ and $I_{i_{min}}$ represent the maximum and minimum of the m grayvalues in the circle, $I_{i_{avg}}$ is the average grayvalue in the circle, and k is a proportional coefficient. The algorithm recalculates the threshold per frame by (16). $I_{i_{max}}$, $I_{i_{min}}$ and $I_{i_{avg}}$ are statistics, and the algorithm performs a linear computation. In the actual operation, we extract 1000 corners at eight-scale levels with a scale factor of 1.2 from indoor scenes. In addition, the orientation of the ORB descriptor is computed on the retained FAST corners. The ORB descriptor is used in all feature matching and loop closure searching.

For line features, LSD [24] algorithm is used to extract line segments. LSD is an $O(n)$ line segment detector with high precision and repeatability, where n is the number of pixels in the image. An illustration of LSD algorithm, which extracts line features in a public dataset and real scene is shown in Fig. 4.

B. Point and Line Association

The similarity between ORB descriptors can be measured by the Hamming distance with respect to the 256-bit binary vectors. In speed up the process, the FLANN algorithm is used to associate them [35]. The FLANN algorithm will yield some mismatches. Therefore, these mismatches are removed using a standard ratio-test method [23].

For line association, we augment line segments with a binary descriptor provided by the Line Band Descriptor (LBD) method [36], which allows for effective line matching between lines based on their local appearance and geometric constraints. LBD is robust against image artifacts while preserving computational efficiency.

C. Pose Solver

In this subsection, in order to get an accurate pose estimation from the point and line correspondences, we proposed a novel point and line-based pose solver.

A flowchart of our pose solver is shown in Fig. 5. First, we use the solvePnPRansac function (Available in the OpenCV library) to process point correspondences, which can produce a set of inliers with respect to points and an initial value for the relative motion matrix. Then, we use the LBIR algorithm

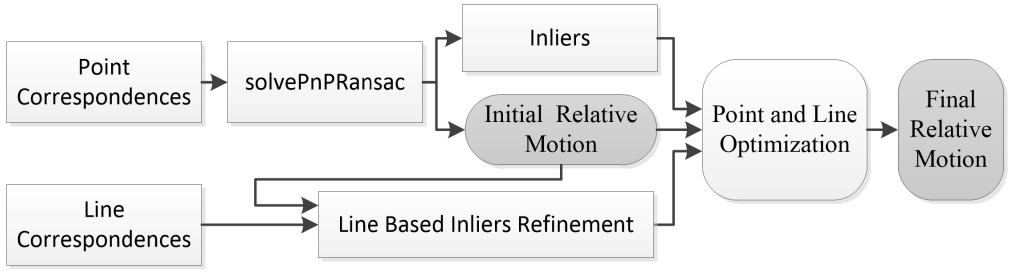

Fig. 5. Flowchart of the pose solver.

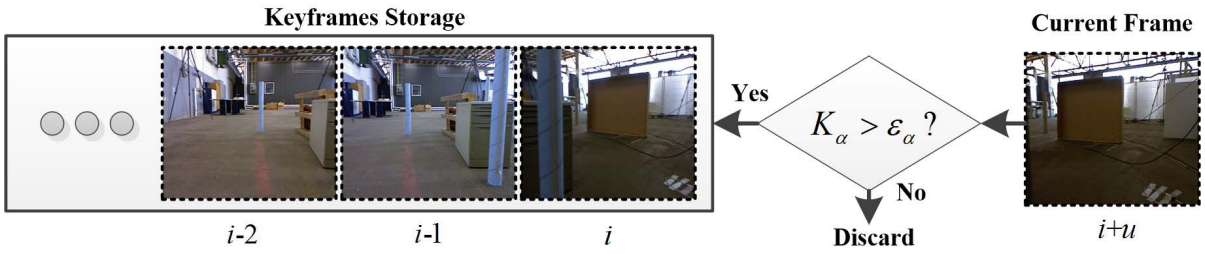

Fig. 6. Illustration of the keyframe decision. The step determines whether the current frame is inserted into the system as a new keyframe.

(See IV-A) to cull outliers, which can produce a set of inliers with respect to lines. Note that the initial value provided by solvePnPRansac is used to accelerate the process. Finally, the initial relative motion is optimized by a point and line optimization algorithm (See IV-B).

VI. Local Mapping

In this section, we introduce our *local mapping* thread. A line solution also create a local bundle adjustment (BA) problem. Therefore, we extend the point and line-based optimization model to local BA (See VI-C).

A. Keyframe Decision

To determine when to save a new keyframe in the keyframes storage, we followed the works of [6], which is different from the strategy in ORB SLAM2 and employs the uncertainty of relative motion estimation. Thus, we transform the uncertainty from the covariance into a scalar $h(\xi)$, called *entropy*, $h(\xi)$, which can be computed by

$$h(\xi) = 3(1 + \log(2\pi)) + 0.5log(|\Sigma_\xi|) \tag{17}$$

where $\xi \in se(3)$ is the 6D vector of the camera motion between two adjacent frames, Σ_ξ denotes the covariance of the ξ, approximated by the inverse of the Hessian of the cost function in the last iteration.

Then, as shown in Fig. 6, given the most recent keyframe K_i, its first consecutive keyframe K_{i-1}, and the current frame F_{i+u}, we evaluate the ratio K_α, which can be given by

$$K_\alpha = \frac{h\left(\xi_{i,i+u}\right)}{h\left(\xi_{i,i+1}\right)} \tag{18}$$

If K_α is less than a pre-established threshold ε_α, the frame F_{i+u} will be inserted into keyframes storage. Note that smaller value means easier policy for maintaining a keyframes storage, which will produces more computation complexity in local BA and loop closure. In this work, the value of ε_α is set to 0.8.

B. Inset Local Keyframe

The local BA builds on a graph optimization [9]. When system selects a keyframe K_i, it is inserted into a local map by adding a new node and updating the edges as a result of shared landmarks with other keyframes. The map scale affects computation. Therefore, it is essential to control the size of a map including numbers of nodes and edges [10].

In our system, once the current frame is determined to be a keyframe, it is inserted into the system with the following information:

- An index for the keyframe;
- Camera pose parameters;
- 3D landmarks including points and lines.

Thus, a keyframe is a node that includes index, pose parameter, and landmarks. Subsequently, we perform data association between the keyframes, and all associations are defined as edges that connect nodes. As a result, local BA is equal to optimizing nodes under edge constraints.

C. Local Bundle Adjustment With Points and Lines

After inserting the keyframe, the camera pose parameters $T_w^c = [R_w^c|t]$ are optimized in the local map with a BA strategy in which constraint T_w^c is in the $se(3)$ group [42]:

$$T_w^c = \exp\left(\xi^\wedge\right) = \begin{bmatrix} R_w^c & t \\ 0^T & 1 \end{bmatrix}, \quad \xi \epsilon se(3) \epsilon \mathcal{R}^6 \tag{19}$$

where $\xi^\wedge$ denotes the anti-symmetric operation that transforms a vector to a skew-symmetric. Differing from other point feature-based SLAM methods (e.g., RGBD SLAM-v2 [9], ORB SLAM2 [10]), we propose a specific cost function to optimize the local map by local BA that combines point and line constraints.

First, for point features, let X_j be the j-th point of the map. For the i-th keyframe, this point can be projected onto π_c, given an observation x_{ij} that denotes the observation of the j-th point in the i-th keyframe, thus, an error function of the point constraint be given by

$$ep_{ij} = x_{ij} - \tilde{x}_{ij} \tag{20}$$

where $\tilde{x}_{ij} = K_c T_i X_j$ is provided by (1) and T_i is the specific pose of the i-th keyframe.

Subsequently, for line features, following the expression of (13), let L_j be the j-th endpoint of the line in the local map. An error function of line constraint can be given by

$$el_{ij} = l_{ij} - \tilde{l}_{ij} \tag{21}$$

where $\tilde{l}_{ij} = \lambda^T K_c T_i L_j$ can be computed by (10).

Finally, we can build a unified cost function that integrates the point and line reprojection errors as

$$F(T) = \sum\nolimits_{i,j} \rho(ep_{ij}^T \Omega_{ij}^{p-1} ep_{ij} + el_{ij}^T \Omega_{ij}^{l-1} el_{ij}) \tag{22}$$

where ρ is the Huber robust cost function, Ω_{ij}^p and Ω_{ij}^l are the covariance matrices associated with the points and line endpoints that are detected, and $T = (T_1^T \dots T_i^T \dots T_n^T)$ is a vector of camera poses.

VII. Loop Closure and Mapping

In this section, we introduce the *loop closure and mapping* thread, which is built on the work of RGBD SLAM-v2 [9] ORB SLAM2 [10], and Bags of Words (BoWs) [37].

A. Loop Closure Detection

This system denotes a keyframe as a set of specific visual words, which is a discretization of the descriptor space (also namely visual vocabulary). The detection of loop closures involves both to find an image similar to the one being currently processed and to estimate a relative pose change between them. The vocabulary is created offline with the ORB descriptors extracted from a large set of images, this system adopts a large and trained vocabulary based on ORB, provided by [10]. Only point features are used for loop detection because matching lines across the whole map is too expensive computationally, which makes system not reach real time.

First, this system computes similarity score between the bag of words vector of K_i and all its neighbors in the keyframes storages, the score can be obtained from [37]. Then, the system queries recognition database, and selects the keyframe with the highest score as a loop closure.

B. Loop Closure Correction

Loop closure correction is performed by a full BA strategy, its form is similar to local BA. The difference is that full BA is a larger optimized process under a global constraint. When system finds a correct loop closure, a transformation matrix is computed between current frame and corresponding keyframe by the proposed pose solver (See V-C).

In practice, to compute a globally consistent trajectory, we optimize the pose graph using the g2o framework [38], which minimizes a nonlinear error function that can be represented as a graph, where error function is the same form as (22). Once finished, a global BA is incorporated to achieve the optimal solution in a separate thread.

C. Map Representation

So far, the system can output a globally consistent trajectory. Therefore, using this trajectory, we can project the original point measurements into a common coordinate frame, thus a point cloud is created. Note that this type of point cloud can directly applied for semantic segmentation or recognition based on 3D point cloud [39], such as the popular deep learning method PointNet [40].

VIII. Experiments

The proposed system was implemented in C++ and its performance was tested in terms of accuracy, robustness, and computing efficiency. A quantitative evaluation of the proposed SLAM system is presented using the well-known indoor public dataset *TUM* [41], and the mapping effort is tested in a real scenario. All of our experiments were performed on a desktop computer (Intel Core i7-8700K CPU @3.70GHz). The software utilized in our evaluation includes Ubuntu, OpenCV, Sophus, Eigen, g2o.

A. TUM Benchmark Datasets

In this section, we compare the proposed system and state-of-the-art solutions using the *TUM* datasets, a well-known RGB-D benchmark to evaluate SLAM system. The dataset consists of many sequences recorded with a Microsoft Kinect RGB-D camera, and provides low-quality images (e.g., rolling shutter, motion blur, and low texture). As a result, many excellent solutions have adopted the *TUM* dataset as their benchmark dataset, such as RGBD SLAM-v2 [9], ORB SLAM2 [10], PL-SLAM [11], and so on. In addition, we adopted the same feature number threshold (1000 points).

1) Absolute Keyframe Trajectory Errors: First, in order to evaluate the localization accuracy, we compare our system against current state-of-the-art visual SLAM solutions, e.g., RGBD SLAM-v2, PL-SLAM, ORB SLAM2, PTAM [5], LSD SLAM [7] and ElasticFusion [4].

The metric used for the comparison is the Absolute Keyframe Trajectory Error (AKTE), provided by the evaluation script of the *TUM* dataset benchmark. The Root-Mean-Square Errors (RMSE) of the AKTE directly measures the difference between the true and estimated trajectory points. For a keyframe trajectory estimation $\hat{X} = \{\hat{x}_1 \dots \hat{x}_n\}$, and the corresponding ground truth X, the value of RMSE can be computed by

$$RMSE_{AKTE} = \sqrt{\frac{1}{n} \sum\nolimits_{i=1}^{n} \left\| trans\left(\hat{x}_i\right) - trans\left(x_i\right) \right\|^2}. \tag{23}$$

Before computing the error, all trajectories are aligned using a similarity warp except for RGBD SLAM-v2 [9] which is aligned using a rigid body transformation.

The results are summarized in Table II. These statistics were extracted from [4], [10], [11], and our own experiments. The specific values are calculated using the median values over 5 executions for each sequence. "$*$" indicates that the algorithm did not provide the data in their paper; "X" means the tracking is lost at some point or a significant portion of the sequence is not processed by the system.

TABLE II

KeyFrame LOCALIZATION ERROR COMPARISON IN THE TUM RGB-D BENCHMARK ABSOLUTE KeyFrame TRAJECTORY (AKE) RMSE [cm]

Sequences	Ours	PL-SLAM	ORB SLAM2	PTAM	LSD-SLAM	RGBD SLAM-v2	ElasticFusion
fr1_xyz	1.16	1.21	1.38	**1.15**	9.00	1.34	*
fr1_floor	6.22	7.59	8.71	X	38.07	**3.51**	*
fr1_desk	2.03	*	**1.69**	X	10.65	2.58	2.0
fr2_xyz	0.40	0.43	0.54	**0.2**	2.15	2.61	1.1
fr2_desk	0.93	*	**0.88**	X	4.57	9.50	7.1
fr2_desk_person	2.68	**1.99**	5.95	X	31.73	6.97	*
fr2_360_kidnap	3.26	3.92	4.99	**2.63**	X	393.3	*
fr2_pioneer_slam	**4.3**	*	10.1	X	X	156.9	*
fr2_large_with_loop	**10.2**	*	15.1	X	X	4699.8	*
fr3_long_office	**1.86**	1.97	4.05	X	38.53	*	1.7
fr3_nstr_tex_far	**2.74**	X	X	34.74	18.31	X	*
fr3_nstr_tex_near	2.10	2.06	1.74	2.74	7.54	*	**1.6**
fr3_nstr_not_near_wl	**7.75**	X	X	42.2	X	*	*
fr3_str_tex_far	**0.92**	0.89	0.98	0.93	7.95	*	*
fr3_str_tex_near	**1.03**	1.25	1.54	1.04	X	*	*
fr3_sit_xyz	**0.07**	**0.07**	0.08	0.83	7.73	*	*
fr3_walk_xyz	1.60	**1.54**	1.64	X	12.44	*	*
fr3_walk_halfsph	**1.58**	1.60	2.09	X	X	*	*

The statistics were extracted from [4], [10-11] and real reproduction experiments. Median over 5 executions for each sequence. The trajectories were aligned with 7DoF with the ground truth before computing the ATE error with the script provided by the benchmark [41]. "*" means that the algorithms do not provide the data in their paper. "X" means that the tracking is lost at some point or a significant portion of the sequence is not processed by the system.

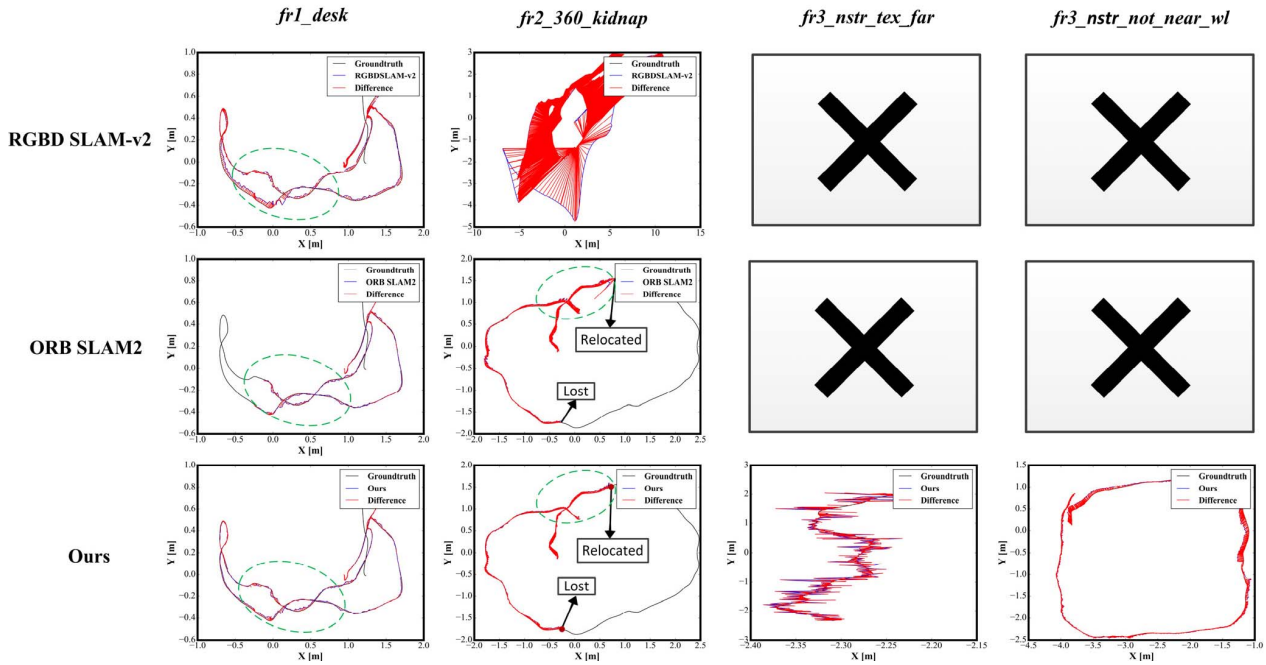

Fig. 7. A motion trajectory comparison of RGBD SLAM-v2, ORB SLAM2 and our method on four representative datasets *fr1_desk*, *fr2_360_kidnap, fr3_nstr_tex_far,* and *fr3_nstr_not_near_wl.* The black and blue lines represent the true and estimated trajectory, the red line is the difference between the true and estimated value. Our system can operate in challenging scenes (*fr3_nstr_tex_far,* and *fr3_nstr_not_near_wl*), while RGBD SLAM-v2 and ORB SLAM2 fail.

In Table II, PTAM had the best accuracy in three sequences. However, this system was not reliable, as tracking was lost in 9 out of all 18 sequences. RGBD SLAM-v2 yielded the best value in a sequence. ORB SLAM2 shows high accuracy and reliability but lost tracking in *fr3_nstr_tex_far* and *fr3_nstr_not_near_wl*, the two sequences with nearly less structured images, and PL-SLAM also failed.

Note that our system performed the best location accuracy in 9 out of all 18 sequences, while ORB SLAM2 only achieves the best location accuracy in 2 out of all 18 sequences. Therefore, we can conclude that the SLAM system using points and lines improves the trajectory accuracy of the solutions depending on point correspondences (e.g., PTAM ORB SLAM2, and RGBD SLAM-v2) in the vast majority of sequences.

2) Relative Pose Errors and Trajectory: The performance of our system was compared with other solutions in terms of Relative Pose Errors (RPE). RPE measures the errors in the relative motion between pairs of timestamps, including RMSE, mean, median, standard deviation, min, max of translational and rotative errors. The values can be obtained using the evaluation script provided by benchmark *TUM toolkit.*

TABLE III
RELATIVE POSE ERROR (RPE*) IN REPRESENTED DATASETS

Methods	Datasets	Translational Error[cm]						Rotative Error[deg]					
		RMSE	Mean	Median	Std.	Min	Max	RMSE	Mean	Median	Mtd.	Min	Max
RGBD SLAM-v2		3.33	2.90	2.62	1.61	0.19	9.75	1.60	1.38	1.22	0.79	0.09	6.48
ORB SLAM2	*fr1_desk*	**2.04**	**1.74**	**1.47**	**1.06**	**0.00**	**6.44**	**1.28**	**1.14**	**1.01**	**0.59**	**0.00**	**3.40**
Ours		2.21	1.84	1.57	1.24	0.09	7.75	1.41	1.04	0.70	0.06	0.06	4.57
RGBD SLAM-v2		229	125	63	192	0.00	1055	35.89	17.79	7.88	31.17	0.00	179.
ORB SLAM2	*fr2_360_kidnap*	7.38	5.02	3.53	5.41	0.00	67.6	8.34	6.94	5.94	4.62	0.00	5.39
Ours		**3.98**	**3.34**	**2.77**	**2.15**	**0.00**	**16.8**	**0.62**	**0.56**	**0.53**	**0.26**	**0.00**	**1.52**
RGBD SLAM-v2		X	X	X	X	X	X	X	X	X	X	X	X
ORB SLAM2	*fr3_nstr_tex_far*	X	X	X	X	X	X	X	X	X	X	X	X
Ours		**3.38**	**2.92**	**2.46**	**1.70**	**0.14**	**1.10**	**0.82**	**0.70**	**0.61**	**0.42**	**0.03**	**2.66**
RGBD SLAM-v2		X	X	X	X	X	X	X	X	X	X	X	X
ORB SLAM2	*fr3_nstr_not_near_wl*	X	X	X	X	X	X	X	**X**	X	X	X	X
Ours		**3.46**	**2.32**	**1.59**	**2.56**	**0.14**	**15.90**	**1.61**	**1.26**	**0.99**	**1.00**	**0.07**	**7.51**

RPE*: TUM benchmark provides a standard evaluation tool based on Python to compute the RPE errors.
"X" means that the tracking is lost at some point or a significant portion of the sequence is not processed by the system.
fr1_desk, fr2_360_kidnap, *fr3_nstr_tex_far* and *fr3_nstr_not_near_wl* include 542, 1252, 420 and 1058 pairs of relative pose, respectively.

The well-known RGBD SLAM-v2 and ORB SLAM2 were selected as reference objects since they are both state-of-the-art solutions with an RGB-D camera, and open-source codes are provided.

We implemented these tests in four representative sequences: *fr1_desk*, *fr2_360_kidnap*, *fr3_nstr_tex_far* and *fr3_nstr_not_ near_wl*. The first is a common scene, the second includes a challenging relocation episode with a long run. The last two sequences represent challenging low-texture scenes.

The results are summarized in Table III, "X" means that tracking is lost at some point or a significant portion of the sequence is not processed by the system. Our system achieves comparable accuracy to ORB SLAM2 in*fr1_desk*, and the best accuracy in the other three sequences.

For visible comparison, we further plotted the finished motion trajectory of the three systems in the four sequences. Fig. 7 provides the motion trajectory comparison. The black and blue lines represent the true and estimated trajectory, the red line is the difference between the true and estimated value. Our system can operate in challenging scenes (*fr3_nstr_tex_far*, and *fr3_nstr_not_near_wl*), while RGBD SLAM-v2 and ORB SLAM2 are prone to failure.

3) Computation Time Analysis: Finally, we investigate the computation performance of our proposed system. Table IV provides a statistic of the computation consumption with respect to *tracking*, *local mapping*, *loop closure* and average frames per second (FPS) in *fr2_large_with_loop*, which is a common scene including a loop closure. RGBD SLAM-v2 only provides the total data since its framework is different from our system, which consists of a front end and back end [9]. Note that the system does not implement local mapping and loop closure per frame, the former is implemented when the system is inserted into a new keyframe; the latter is implemented when the system finds a loop closure.

Table IV shows that RGBD SLAM-v2 takes the most time, because it adopts SIFT as its feature extractor, which is the well-known representation of high precision and consumption. ORB SLAM2 requires the last amount of time. Compared with ORB SLAM2, our system operates at a lower FPS with higher accuracy because our method merges line features and point feature.

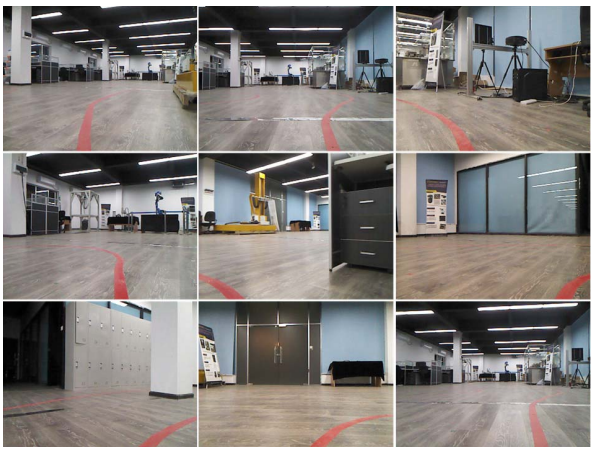

Fig. 8. Example of images captured by Kinect1 in the real world.

TABLE IV
AVERAGE ELAPSED TIME [MS] OF EACH STEP IN *Fr2_Large_With_Loop*

Methods	Tracking	Local Mapping	Loop Closure	Avg. FPS
RGBD SLAM-v2	X	X	X	16.5HZ
ORB SLAM2	20.12	50.11	145.65	38.7HZ
Ours	30.23	71.63	162.73	29.4HZ

B. Real Scenario

In this subsection we construct our experiments in real scenario by visual and quantitative comparisons. The TurlteBot is used for the mobile platform, which is a low-cost, personal robot kit with open-source software. The Turltebot was equipped with the camera and moves following a designed trajectory to capture images.

1) Visual Experiments: Visual experiments are performed by point cloud map. Fig. 8 shows examples of images captured by a typical RGB-D camera, Kinect 1. This scene includes various situations such as rolling shutter, motion blur, and low texture, which is a challenge for visual SLAM.

Fig. 9 provides a comparison of 3D point clouds between RGBD SLAM-v2 and our system from three perspectives. Visually, RGBD SLAM-v2 yields a big drifting in the white

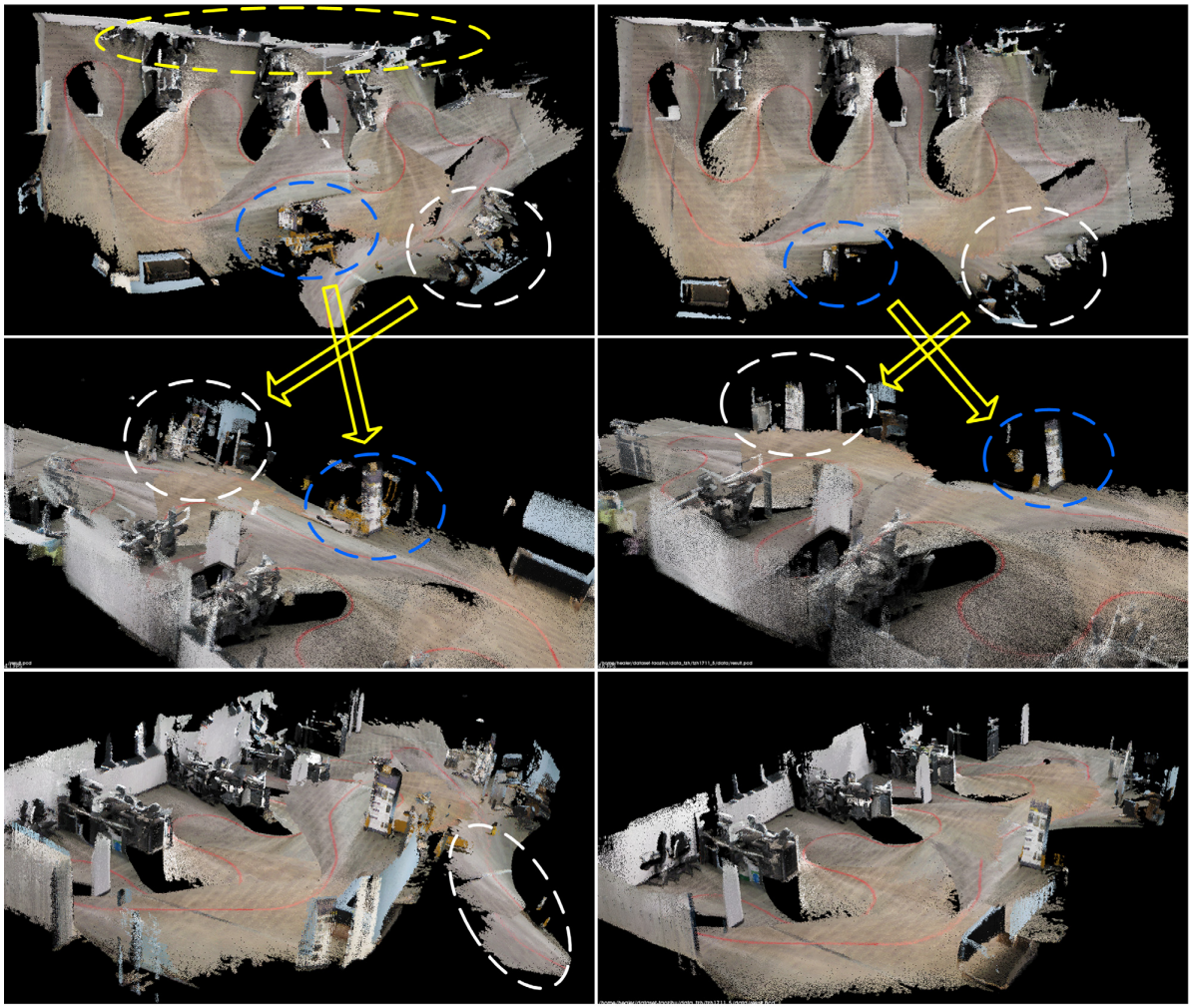

Fig. 9. Comparison of 3D point clouds from our method and RGBD SLAM-v2. The left is generated by RGBD SLAM-v2; the right is generated by our system. By comparing with a real environment, RGBD SLAM-v2 yields a large amount of drifting in the white circle and blue circle, furthermore, in the yellow circle, this system produces a curved wall (The wall should be straight). Generally, our system outputs a more subtle and complete point map.

circle and blue circle. Furthermore, in the yellow circle, RGBD SLAM-v2 produces a wall that is more curved than from our system (The wall should be straight). Generally, our system outputs more subtle and complete point map. Therefore, our system is more robust and reliable.

2) Quantitative Experiments: Quantitative experiments are performed by closure errors. There is no ground truth in the real scenario, therefore, we use closure error as quantitative evaluation, which can be calculated to evaluate the accuracy without using loop closure detection function. The Turltebot moving path is designed to start at the starting position and then go back to the starting position to calculate the closure error, as Fig. 10 shows.

The right column in the Fig. 10 provides motion trajectories comparison of RGBD SLAMv2, ORB SLAM2 and ours. Their closure errors are 36.85 cm (RGBD SLAMv2), 15.74 cm (ORB SLAM2) and 9.37 cm (Ours), ours system has the best closure errors accuracy. Therefore, our system is more reliable.

C. Results and Discussion

In this subsection, we discuss the experimental results in challenging scenes.

1) Low Texture: In low texture, it is common for system to not extract enough features, which will directly lead to failure.

"fr3-sequences" in *TUM* (the last 9 sequences in Table II) are designed to test the robustness of a system in low-texture scenes, and our system has the best RMSE accuracy in 7 out of 9 sequences. In addition, from the two right columns in Fig. 7 and Table II, our system even finished relatively complete trajectories in extremely challenging sequences, e.g., *fr3_nstr_tex_far* and *fr3_nstr_not_near_wl,* including plenty of nearly texture-less images, while RGBD SLAM-v2, ORB SLAM2, PTAM and PL-SLAM all lost track. Therefore, our system is more robust than these solutions.

2) Long Term Runs: "fr2-sequences" have the longest trajectories with difficult conditions in *TUM*, in which, *fr2_pioneer_slam* and *fr2_large_ with_loop* are the sequences with the two longest trajectories and time. Our system has the best RMSE accuracy for these sequences (See Table II). *fr2_360_kidnap* includes a human-designed relocation process, and our system can relocate and has the better RPE errors than RGBD SLAM-v2 and ORB SLAM2 in this sequence (See Table II and Fig. 7). In *fr2_360_kidnap* (the second column

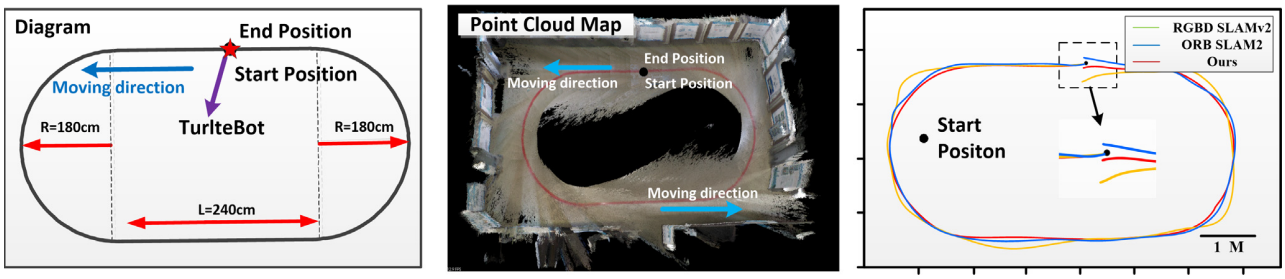

Fig. 10. Quantitative experiments in real scenario. Left column is a diagram that the moving path is designed to start at the starting point and then go back to the starting point to calculate the closure error. Middle column is the point cloud map that is generated by our system. Right column provides motion trajectories of RGBD SLAMv2, ORB SLAM2 and ours.

in Fig. 7), our system lost track at approximately 15 s and relocated at approximately 30 s, which proves that our method can perform relocation by the loop closure module when robot lost tracking. Therefore, our system is more reliable and robust than the other two solutions.

IX. Conclusion

In this paper, we present a robust RGBD SLAM system using points and lines that is applicable for real-time indoor mobile robot pose estimation. By exploring the line features application, we propose a novel pose solver to determine pose from point and line correspondences. Experiments proved our system can improve upon the robustness and accuracy of other systems that only use points, e.g., PTAM, ORB SLAM2, RGBD SLAM-v2. In addition, our system produce a more robust and reliable 3D map than RGBD SLAM-v2 in a large real-world experiment. Notably, our system can operate in some extremely challenging environments while other solutions all lose tracking, such as for *fr3_nstr_tex_far*.

In future works, we plan to focus on improving two aspects of our current works: 1) The point cloud is highly redundant and requires vast storage resources; therefore, an efficient representation of the point cloud can be studied; 2) the advantages of a line feature have still not been fully exploited; we plan to solve the problem of line feature occlusion and the uncertainty problem of the endpoints for extracted lines.

References

[1] K. D. Sharma, A. Chatterjee, and A. Rakshit, "Experimental study II: Vision-based navigation of mobile robots," in *Intelligent Control*. Singapore: Springer, 2018, pp. 243–280.

[2] T. Mouats, N. Aouf, L. Chermak, and M. A. Richardson, "Thermal stereo odometry for UAVs," *IEEE Sensors J.*, vol. 15, no. 11, pp. 6335–6347, Nov. 2015.

[3] C. Cadena *et al.*, "Past, present, and future of simultaneous localization and mapping: Toward the robust-perception age," *IEEE Trans. Robot.*, vol. 32, no. 6, pp. 1309–1332, Dec. 2016.

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

[5] G. Klein and D. Murray, "Parallel tracking and mapping for small AR workspaces," in *Proc. 6th IEEE ACM Int. Symp. Mixed Augmented Reality*, Nov. 2007, pp. 225–234.

[6] C. Kerl, J. Sturm, and D. Cremers, "Dense visual SLAM for RGB-D cameras," in *Proc. IEEE/RSJ Int. Conf. Intell. Robots Syst.*, Nov. 2013, pp. 2100–2106.

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

[8] M. Labbé and F. Michaud, "Appearance-based loop closure detection for Online large-scale and long-term operation," *IEEE Trans. Robot.*, vol. 29, no. 3, pp. 734–745, Jun. 2013.

[9] F. Endres, J. Hess, J. Sturm, D. Cremers, and W. Burgard, "3-D mapping with an RGB-D camera," *IEEE Trans. Robot.*, vol. 30, no. 1, pp. 177–187, Feb. 2014.

[10] R. Mur-Artal and J. D. Tardós, "ORB-SLAM2: An open-source SLAM system for monocular, stereo, and RGB-D cameras," *IEEE Trans. Robot.*, vol. 31, no. 5, pp. 1147–1163, Oct. 2017.

[11] A. Pumarola, A. Vakhitov, A. Agudo, A. Sanfeliu, and F. Moreno-Noguer, "PL-SLAM: Real-time monocular visual SLAM with points and lines," in *Proc. IEEE Int. Conf. Robot. Automat. (ICRA)*, May/Jun. 2017, pp. 4503–4508.

[12] R. Gomez-Ojeda, F. Moreno, D. Zuñiga-Noël, D. Scaramuzza, and J. Gonzalez-Jimenez, "PL-SLAM: A stereo SLAM system through the combination of points and line segments," *IEEE Trans. Robot.*, vol. 35, no. 3, pp. 734–746, Jun. 2019. doi: 10.1109/TRO.2019.2899783.

[13] Q. Yu, J. Xiao, H. Lu, and Z. Zheng, "Hybrid-residual-based RGBD visual odometry," *IEEE Access*, vol. 6, pp. 28540–28551, 2018.

[14] H. Yu, Q. Fu, Z. Yang, L. Tan, W. Sun, and M. Sun, "Robust robot pose estimation for challenging scenes with an RGB-D camera," *IEEE Sens. J.*, vol. 19, no. 6, pp. 2217–2229, Mar. 2018. doi: 10.1109/JSEN.2018.2884321.

[15] G. Lan *et al.*, "Development of UAV based virtual reality systems," in *Proc. IEEE Int. Conf. Multisensor Fusion Integr. Intell. Syst. (MFI)*, Sep. 2016, pp. 481–486.

[16] G. Lan, J. Liang, G. Liu, and Q. Hao, "Development of a smart floor for target localization with Bayesian binary sensing," in *Proc. IEEE 31st Int. Conf. Adv. Inf. Netw. Appl. (AINA)*, Mar. 2017, pp. 447–453.

[17] 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.

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

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

[20] J. Engel, J. Sturm, and D. Cremers, "Semi-dense visual odometry for a monocular camera," in *Proc. IEEE Int. Conf. Comput. Vis. (ICCV)*, Dec. 2013, pp. 1449–1456.

[21] A. J. Davison, "MonoSLAM: Real-time single camera SLAM," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 1, no. 6, pp. 1052–1067, Jun. 2007.

[22] A. S. Huang *et al.*, "Visual odometry and mapping for autonomous flight using an RGB-D camera," in *Robotics Research*. Cham, Switzerland: Springer, 2017, pp. 235–252.

[23] E. Rublee, V. Rabaud, K. Konolige, and G. Bradski, "ORB: An efficient alternative to SIFT or SURF," in *Proc. Int. Conf. Comput. Vis. (ICCV)*, Nov. 2011, pp. 2564–2571.

[24] R. G. von Gioi, J. Jakubowicz, J. Morel, and G. Randall, "LSD: A fast line segment detector with a false detection control," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 32, no. 4, pp. 722–732, Apr. 2010.

[25] G. Zhang, J. H. Lee, J. Lim, and I. H. Suh, "Building a 3-D line-based map using stereo SLAM," *IEEE Trans. Robot.*, vol. 31, no. 6, pp. 1364–1377, Dec. 2015.

[26] H. Zhou, D. Zou, L. Pei, R. Ying, P. Liu, and W. Yu, "StructSLAM: Visual SLAM with building structure lines," *IEEE Trans. Veh. Technol.*, vol. 64, no. 4, pp. 1364–1375, Apr. 2015.

[27] D. Van-Opdenbosch and E. Steinbach, "Collaborative visual SLAM using compressed feature exchange," *IEEE Robot. Autom. Lett.*, vol. 4, no. 1, pp. 57–64, Jan. 2019.
[28] C. Xu, L. Zhang, L. Cheng, and R. Koch, "Pose estimation from line correspondences: A complete analysis and a series of solutions," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 39, no. 6, pp. 1209–1222, Jun. 2017.
[29] R. Wang, K. Di, W. Wan, and Y. Wang, "Improved point-line feature based visual SLAM method for indoor scenes," *Sensors*, vol. 18, no. 10, p. 3559, Oct. 2018. doi: 10.3390/s18103559.
[30] S. Li, T. Zhang, X. Gao, D. Wang, and Y. Xian, "Semi-direct monocular visual and visual-inertial SLAM with loop closure detection," *Robot. Auton. Syst.*, vol. 112, pp. 201–210, Feb. 2019. doi: 10.1016/j.robot.2018.11.009.
[31] T. Lee, C. Kim, and D.-I. D. Cho, "A monocular vision sensor-based efficient SLAM method for indoor service robots," *IEEE Trans. Ind. Electron.*, vol. 66, no. 1, pp. 318–328, Jan. 2018.
[32] F. Moreno-Noguer, V. Lepetit, and P. Fua, "Accurate non-iterative O(n) solution to the PnP problem," in *Proc. IEEE 11th Int. Conf. Comput. Vis. (ICCV)*, Oct. 2007, pp. 1–8.
[33] A. Vakhitov, J. Funke, and F. Moreno-Noguer, "Accurate and linear time pose estimation from points and lines," in *Proc. Eur. Conf. Comput. Vis. (ECCV)*. Cham, Switzerland: Springer, Sep. 2016, pp. 583–599.
[34] L. Ferraz, X. Binefa, and F. Moreno-Noguer, "Very fast solution to the PnP problem with algebraic outlier rejection," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit. (CVPR)*, Jun. 2014, pp. 501–508.
[35] M. Muja and D. Lowe, "Fast matching of binary features," in *Proc. IEEE Conf. Comput. Robot Vis. (CRV)*, 2012, pp. 404–410.
[36] L. Zhang and R. Koch, "An efficient and robust line segment matching approach based on LBD descriptor and pairwise geometric consistency," *J. Visual Commun. Image Represent.*, vol. 24, no. 7, pp. 794–805, Oct. 2013.
[37] D. Galvez-López and J. D. Tardos, "Bags of binary words for fast place recognition in image sequences," *IEEE Trans. Robot.*, vol. 28, no. 5, pp. 1188–1197, Oct. 2012.
[38] R. Kümmerle, G. Grisetti, H. Strasdat, K. Konolige, and W. Burgard, "G^2o: A general framework for graph optimization," in *Proc. IEEE Int. Conf. Robot. Autom. (ICRA)*, May 2011, pp. 3607–3613.
[39] L. Ge, Z. Ren, and J. Yuan, "Point-to-point regression pointnet for 3D hand pose estimation," in *Proc. Eur. Conf. Comput. Vis. (ECCV)*. Cham, Switzerland: Springer, 2018, pp. 489–505.
[40] C. R. Qi, H. Su, K. Mo, and L. J. Guibas, "Pointnet: Deep learning on point sets for 3d classification and segmentation," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit. (CVPR)*, Jul. 2017, pp. 77–85.
[41] 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.*, Oct. 2012, pp. 573–580.
[42] C. Xu, L. N. Govindarajan, Y. Zhang, and L. Cheng, "Lie-X: Depth image based articulated object pose estimation, tracking, and action recognition on lie groups," *Int. J. Comput. Vis.*, vol. 123, no. 3, pp. 454–478, Jul. 2017.
[43] D. G. Lowe, "Object recognition from local scale-invariant features," in *Proc. 7th IEEE Int. Conf. Comput. Vis.*, vol. 2, Sep. 1999, pp. 1150–1157.
[44] H. Bay, A. Ess, T. Tuytelaars, and L. Van Gool, "Speeded-up robust features," *Comput. Vis. Image. Understand.*, vol. 110, no. 3, pp. 346–359, 2008.

Qiang Fu received the B.S. degree from Hunan Normal University, Changsha, China, in 2015. He is currently pursuing the Ph.D. degree with Hunan University, Changsha.

His research interests include visual SLAM, computer vision, and robotics.

Hongshan Yu received the B.S., M.S., and Ph.D. degrees in control science and technology in electrical and information engineering from Hunan University, Changsha, China, in 2001, 2004, and 2007, respectively.

From 2011 to 2012, he was a Post-Doctoral Researcher with the Laboratory for Computational Neuroscience, University of Pittsburgh, USA. He is currently a Professor with Hunan University and Associate Dean of the National Engineering Laboratory for Robot Visual Perception and Control. His research interests include autonomous mobile robot and machine vision.

Lihai Lai received the B.E. degree from Anyang Normal University, Anyang, China, in 2015. He is currently pursuing the master's degree with Hunan Normal University, Changsha, China.

His research interests include visual SLAM, mobile robot localization, and computer vision.

Jingwen Wang received the B.E. degree from Hunan University, Changsha, China, in 2017, where she is currently pursuing the master's degree.

Her research interests include visual SLAM and computer vision.

Xia Peng received the B.E. degree in electrical and electronics engineering from the Hunan Institute of Engineering, Xiangtan, China, in 2016. She is currently pursuing the master's degree with Central South University, Changsha, China.

Her research interests include intelligent transportation systems and mobile robot.

Wei Sun received the M.S. and Ph.D. degrees in control science and technology from Hunan University, Changsha, China, in 1999 and 2002, respectively.

He is currently a Professor at Hunan University. His research interests include artificial intelligence, robot control, complex mechanical and electrical control systems, and auto-motive electronics.

Mingui Sun received the B.S. degree in instrumental and industrial automation from the Shenyang Chemical Engineering Institute, Shenyang, China, in 1982, and the M.S. and Ph.D. degrees in electrical engineering from the University of Pittsburgh, Pittsburgh, PA, USA, in 1986 and 1989, respectively.

He is currently a Professor of Neurosurgery, Electrical and Computer Engineering, and Bioengineering. His current research interests include advanced biomedical electronic devices, biomedical signal and image processing, sensors and transducers, and artificial neural networks.