Geometric Primitives Based RGB-D SLAM for Low-texture Environment

Penglei Ji
The State Key Lab of CAD&CG
Zhejiang University
Hangzhou, 310058, China
jpl@zju.edu.cn

Ming Zeng
Software School of
Xiamen University
Xiamen, 361000, China
zengming@xmu.edu.cn

Xinguo Liu
The State Key Lab of CAD&CG
Zhejiang University
Hangzhou, 310058, China
xgliu@cad.zju.edu.cn

ABSTRACT

Visual Simultaneous Localization and Mapping (SLAM) is very important in various applications such as AR, Robotics, etc. A challenging problem in SLAM is the inferior tracking performance in the low-texture environment due to their low-level feature based tactic. This paper presents a novel SLAM system which leverages feature-wise alignment and layout information to handle this challenge. The key idea is to utilize the points, lines and planes with both feature-wise constraints and layout consistency constraints to obtain robust motion estimation. Firstly, we extract the points, lines and planes from color images and depth images. Then, we find some matches between these geometric primitives. Lastly, we propose a unified solution to represent the local primitives information and the layout context among the extracted primitives, thus to stabilize the camera motion estimation. Experiments on TUM datasets demonstrate that our method outperforms other RGB-D SLAM systems in the low-texture environment, and provides comparable results in the richly-textured environment.

CCS CONCEPTS

• **Computing methodologies** → **Reconstruction**; **Matching**;

KEYWORDS

Geometric Primitives, Layout Consistency, Low-texture, RGB-D SLAM

ACM Reference Format:
Penglei Ji, Ming Zeng, and Xinguo Liu. 2018. Geometric Primitives Based RGB-D SLAM for Low-texture Environment. In *CASA 2018: 31st International Conference on Computer Animation and Social Agents, May 21–23, 2018, Beijing, China.* ACM, New York, NY, USA, 7 pages. https://doi.org/10.1145/3205326.3205358

CASA 2018, May 21–23, 2018, Beijing, China

ACM ISBN 978-1-4503-6376-1/18/05…$15.00
https://doi.org/10.1145/3205326.3205358

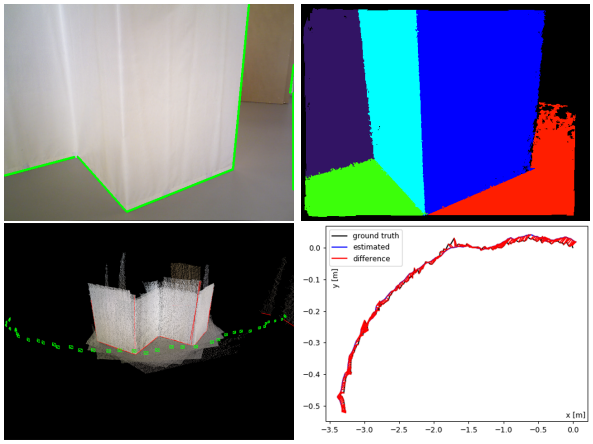


Figure 1: An example of 3D reconstruction in the low-texture TUM dataset (fr3/str-notex-far). Color image with detected lines marked in green (top-left), Plane extraction result (top-right), Map reconstructed with point and line landmarks (bottom-left), Trajectory error compared with the ground-truth (bottom-right).

1 INTRODUCTION

Visual SLAM represents simultaneously tracking the camera pose and reconstructing the 3D environment. There are mainly two kinds of methods in the RGB-D SLAM community: direct-based methods and feature-based methods. Currently, feature-based ORB-SLAM2 [11] becomes the state-of-the-art method, and provides more accurate camera poses than direct-based methods, such as KinectFusion [12].

Though ORB-SLAM2 gains impressive performance in the richly-textured environment, it often fails in the low-texture environment due to the lack of keypoints. Direct-based methods can handle the low-texture environment theoretically, owing to the dense matching procedure. However, they only work well for slow and continuous motion, and are prone to local minimum under the fast and discontinuous motion.

Low-texture challenge can be alleviated by introducing more high-level geometric primitives. Some methods [9, 10, 17] exploit one or some of primitives, e.g. point, line, or plane, in a feature-wise fashion, without investigating their layout relationships to enhance the performance of feature matching. However, layout relationships among the primitives encode rich structure information of the scenes. This is a very strong

cue especially in the low-texture environment, which could be employed to reduce the solution space dramatically, thereby improve the performance of camera tracking.

Based on the above consideration, we propose a novel RGB-D SLAM system which combines geometric primitives and the layout relationships between them into a unified solution to estimate the camera motion. As shown in Fig. 1, in such a low-texture environment, our method recovers the camera trajectory successfully, and outperforms the state-of-the-art ORB-SLAM2 which always fails due to the poor texture.

Our main contributions are as follows:

- Integrate the point features, line features and plane features to enhance the feature-based RGB-D SLAM in the low-texture environment.
- Propose a unified solution which combines all the geometric primitives and the layout relationships between them to estimate the camera motion.
- Demonstrate a real-time RGB-D SLAM system that performs better than existing approaches and builds a global consistent dense map.

2 RELATED WORK

2.1 Direct-based SLAM

Direct-based RGB-D SLAM systems usually start with a good initialization, and register two dense point sets iteratively. The most popular algorithm is the Iterative Closest Point (ICP) [2], and many variants of ICP have been proposed to perform the frame-to-frame registration.

KinectFusion [12] extends the traditional ICP to register a depth map to the depth map generated from a Truncated Signed Distance Function (TSDF). The nearest correspondences of all points are calculated and then used to refine the transformation iteratively. ElasticFusion [19] performs the dense frame-to-model camera tracking and builds a surfel-based environment map. The complexity of this method is proportional to the number of surfels. Zhang *et al.* [22] extended the KinectFusion to fuse the planar regions and nonplanar objects, and produce structured models in real time. Direct-based methods usually rely on the dense matching, and require GPU acceleration for real-time performance. Also, they often suffer from the local minimum problem.

2.2 Feature-based SLAM

In feature-based SLAM, camera motion is estimated with the feature matches, rather than all dense points. The number of features is much smaller than all points, thus reducing the computation cost.

Point feature based. Henry *et al.* [6] extracted the keypoints from a color image, and back-projected them to 3D space using the depth image. The camera motion is estimated from 3 point correspondences and then set as an initialization of ICP. RGB-D SLAM [4] computes the frame-to-frame motion using SIFT feature and ICP algorithm. ORB-SLAM2 [11] uses the ORB features for the camera tracking, mapping and relocalization. The major challenge is that the feature tracking is often corrupted due to the lack of keypoints in the low-texture environment.

Line feature based. Line features are ubiquitous even in the low-texture environment, and provide more geometric information than point features. StructSLAM [23] uses the structure lines as features for localization, and extends the standard Extended Kalman Filter to adopt the lines with a novel parameterization. PLVO [9] fuses the points and lines to form a robust RGB-D odometry, and uses maximum likelihood estimation to compute the camera motion. Pumarola *et al.* [13] and Yang *et al.* [20] both proposed a point and line based monocular SLAM to recover the camera motion.

Plane feature based. Plane is a higher level geometric primitive than point and line, and can benefit the accuracy of camera motion estimation. Taguchi *et al.* [17] proposed a point-plane SLAM which combines the point and plane observations to compute correspondences and estimate the camera motion. Salas-Moreno *et al.* [15] put forward the dense planar SLAM which densely maps an environment with the bounded planes and surfels extracted from depth images. Ma *et al.* [10] proposed CPA-SLAM, an RGB-D SLAM system that only uses the planes for tracking and models the environment with a global plane model. Pop-up SLAM [21] is a monocular system which combines planes with points. It is based on the ground semantic segmentation and 3D model pop-up algorithm.

Motivated by above works, we combine all the geometric primitives and the layout relationships between them to improve the feature-based RGB-D SLAM in the low-texture environment.

3 SYSTEM OVERVIEW

The pipline of our system is shown in Fig. 2. The input of our system is pairs of a color image and a depth image. The outputs are the trajectory of the camera motion and the constructed dense map. We firstly extract all geometric primitives from the input images. Then We find some matches with these features. And we estimate the camera motion using the feature-wise constraints and the layout consistency constraints between primitives. Lastly, we perform a global pose graph optimization. Both the motion estimation and pose graph optimization are based on the g2o [8] framework, an open-source graph-based nonlinear error optimizer.

As shown in Fig. 3, the goal of camera motion estimation is to estimate the 3D rigid transformation T from Frame_n to Frame_{n+1}. For better visualization, we only draw the connections between cameras and line features, and the layout relationships between primitives. Solid lines represent the corresponding primitives in different frames. Dotted red lines represent the layout relationships between primitives, and they should be same in different camera coordinate systems.

The camera motion estimation problem can be formulated as an energy minimization problem. We define the total energy function as a sum of the local feature-wise alignment error $E_{feature}$ and the global layout consistency error E_{layout}.

$$T^* = \arg\min_T \left(E_{feature} + E_{layout}\right) \quad (1)$$

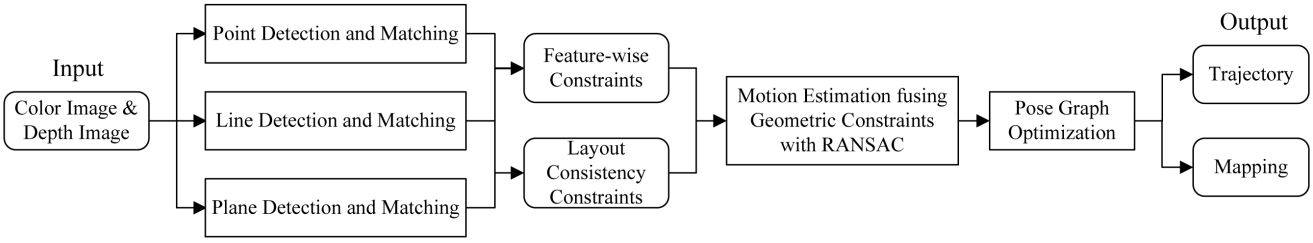


Figure 2: The pipline of our system.

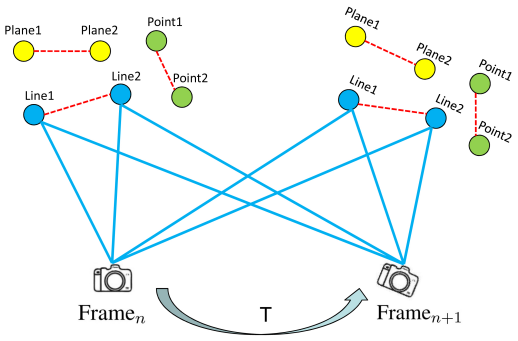


Figure 3: Camera motion estimation problem.

$E_{feature}$ penalizes that the corresponding features that are not of the same element in the world coordinate, and E_{layout} penalizes that the layout relationships between primitives in Frame$_n$ that change in Frame$_{n+1}$.

4 GEOMETRIC PRIMITIVES

4.1 Point detection and matching

We adopt the ORB [14] feature which is rotation invariant and associated with a 256-bits binary descriptor. Compared with SIFT and SURF, ORB gains a good trade-off between accuracy and computation complexity. Grid-based Motion Statistics (GMS) [3] algorithm is applied after brute force matching to obtain the more robust matches.

4.2 Line detection and matching

We use the Edge Drawing method (ED) [1] to detect lines, and choose the Mean Standard deviation Line Descriptor (MSLD) [18] for line matching. Each line segment l_i in 2D image is represented by two endpoints (p_i, q_i) and associated with a MSLD descriptor. To obtain the 3D position of a 2D line segment, a direct method is to back-project the two 2D endpoints to 3D space using the depth image. However, this method is not robust owing to the depth corruption and value ambiguity. To address this problem, we sample n points evenly on the 2D line segment, and then back-project them to the 3D space. Because of the noise and ambiguity, these 3D points may not be co-linear perfectly. We then perform the maximum likelihood estimation to calculate the linear fitting equation of these 3D points and the endpoints of 3D lines.

4.3 Plane detection and matching

Planes contain more high level structure information than points and lines, and are more robust to noise, yielding more accurate registrations. They can also be used to recover the planar structure of the environment.

Plane detection is based on the algorithm described in [5], which uses the Agglomerative Hierarchical Clustering (AHC) for fast plane extraction. The method first initializes a node graph and then performs AHC algorithm to cluster those similar neighbor nodes. For each plane, the equation $ax + by + cz + d = 0$ $s.t.$ $a^2 + b^2 + c^2 = 1$ is calculated by PCA in a constant time operation. We redefine a plane as $\pi = (\mathbf{n}^{\mathrm{T}}, d)^{\mathrm{T}}$, where $\mathbf{n} = (a, b, c)^{\mathrm{T}}$ is the unit normal vector and d is the distance from the plane to the origin.

The plane matches are obtained using the angle between plane normals and the difference between the distance to the origin. Two planes $\pi_1 = (\mathbf{n_1}^{\mathrm{T}}, d_1)^{\mathrm{T}}$ and $\pi_2 = (\mathbf{n_2}^{\mathrm{T}}, d_2)^{\mathrm{T}}$ with $\mathbf{n_1} \cdot \mathbf{n_2} > \cos(\theta)$ and $||d_1 - d_2|| < \delta$ will be matched. We set $\theta = 15^\circ$ and $\delta = 0.05$ in our experiments.

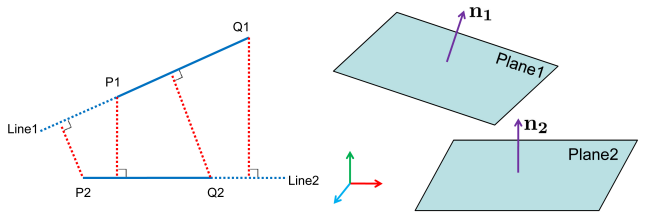
5 MOTION ESTIMATION AND LAYOUT CORRECTION

Denote the RGB-D frame at time k as $F_k = (I_k, D_k)$, where I_k and D_k represent the color image and the depth image respectively. For simplify, we choose the calculation of $T_{21} = [R_{21} \mid t_{21}]$ between $F_1 = (I_1, D_1)$ and $F_2 = (I_2, D_2)$ as an example. Based on the geometric primitives detection and matching introduced in Section 4, we estimate the relative motion from F_1 to F_2 using these features and the layout relationships between them.

The role of layout consistency constraints is de-noising and correcting the layout of geometric primitives to avoid the false matches established in the feature matching procedure.

5.1 Feature-wise alignment

Point reprojection error. Given an RGB-D frame F, we detect a set of keypoints from the color image I, and compute their corresponding descriptor using the ORB feature. Supposing a keypoint $p^i = [u^i,\ v^i]^{\mathrm{T}}$ in color image I and the corresponding depth d^i in depth image D if available, its 3D position P^i in camera coordinate system can be computed



(a) (b)

Figure 4: (a) The distance between 3D lines is defined as the sum of four distances from one endpint to the other line. (b) The distance between planes is defined based on their normals and the distances from plane to the origin.

as follows:

$$P^i := \begin{bmatrix} X^i \\ Y^i \\ Z^i \end{bmatrix} = \begin{bmatrix} (u^i - c_x)d^i/f_x \\ (v^i - c_y)d^i/f_y \\ d^i \end{bmatrix} \tag{2}$$

Here, we define K as the camera calibration matrix, and f_x, f_y, c_x, c_y are the intrinsic parameters, representing the focal length and the principal point of the color camera respectively.

Let $(p_1^i,\ p_2^i), i = 1, \cdots, m$, be the m pairs of 2D point matches in I_1 and I_2 , and $(P_1^i,\ P_2^i), i = 1, \cdots, m$, the corresponding 3D point pairs.

We project the P_1^i to image I_2 and calculate its projection $\widetilde{p_1^i}$ as below:

$$d_2^i \widetilde{p_1^i} = K(R_{21}P_1^i + t_{21}). \tag{3}$$

The reprojection error of point P_1^i in I_2 is defined as:

$$E_{pt}^i = ||\ p_2^i - \widetilde{p_1^i}\ ||^2. \tag{4}$$

Line alignment error. Let $(l_1^i,\ l_2^i), i = 1, \cdots, n$, be the n pairs of 2D line matches in I_1 and I_2, and $(L_1^i,\ L_2^i), i = 1, \cdots, n$, the corresponding 3D line pairs.

We use $P_1^i, Q_1^i \in R^3$ to represent the two 3D endpoints of the line L_1^i in F_1. We transform the line to F_2 coordinate system as $\widetilde{L_1^i}$ whose two endpoints are represented as $\widetilde{P_1^i}$ and $\widetilde{Q_1^i}$:

$$\widetilde{P_1^i} = R_{21}P_1^i + t_{21}, \quad \widetilde{Q_1^i} = R_{21}Q_1^i + t_{21}. \tag{5}$$

As shown in Fig. 4(a), the distance D_{ln} between two 3D lines L_1 and L_2 can be calculated as the sum of the four distances from one endpoint to the other 3D line.

$$\begin{aligned} D_{ln}(L_1, L_2) =& D_{pt2ln}(P_1, L_2) + D_{pt2ln}(Q_1, L_2) \\ &+ D_{pt2ln}(P_2, L_1) + D_{pt2ln}(Q_2, L_1) \end{aligned} \tag{6}$$

where $D_{pt2ln}(P, L)$ is the distance from 3D point P to 3D line L.

The line alignment error for L_1^i and L_2^i can be represented as:

$$E_{ln}^i = D_{ln}(L_2^i, \widetilde{L_1^i}). \tag{7}$$

Plane alignment error. Let $\pi_1^i = (\mathbf{n_1^i}^{\mathrm{T}}, d_1^i)^{\mathrm{T}}$ and $\pi_2^i = (\mathbf{n_2^i}^{\mathrm{T}}, d_2^i)^{\mathrm{T}}, i = 1, \cdots, l$, be the l pairs of corresponding planes in F_1 and F_2. We transform the plane π_1^i to the F_2 coordinate system as $\widetilde{\pi_1^i} = (\widetilde{\mathbf{n_1^i}}^{\mathrm{T}}, \widetilde{d_1^i})^{\mathrm{T}}$:

$$\widetilde{\mathbf{n_1^i}} = R_{21}\mathbf{n_1^i}, \quad \widetilde{d_1^i} = d_1^i - \mathbf{n_2^i}^{\mathrm{T}} t_{21}. \tag{8}$$

As shown in Fig. 4(b), there are two planes $\pi_1 = (\mathbf{n_1}^{\mathrm{T}}, d_1)^{\mathrm{T}}$ and $\pi_2 = (\mathbf{n_2}^{\mathrm{T}}, d_2)^{\mathrm{T}}$ in the same coordinate system. The distance D_{pl} between them is defined as:

$$D_{pl}(\pi_1, \pi_2) = ||\mathbf{n_1} - \mathbf{n_2}||^2 + (d_1 - d_2)^2. \tag{9}$$

The plane alignment error is expressed as:

$$E_{pl}^i = D_{pl}(\pi_2^i, \widetilde{\pi_1^i}). \tag{10}$$

5.2 Layout correction

Layout correction aims to reject the false matches and enhance the robustness of camera motion estimation.

Point-point relationship. This error term L_{pt} penalizes that the layout relationships between points in F_1 that do not remain the same in F_2.

$$\begin{aligned} L_{pt} = &\sum_{i,j} (||P_1^i - P_1^j||^2 - ||P_2^i - P_2^j||^2) \\ &s.t.\ \forall i, j \in [1, m]\ and\ i \neq j. \end{aligned} \tag{11}$$

Line-line relationship. With the definition of distance D_{ln} between two lines in Section 5.1, the line layout consistency error is defined as:

$$\begin{aligned} L_{ln} = &\sum_{i,j} (D_{ln}(L_1^i, L_1^j) - D_{ln}(L_2^i, L_2^j)) \\ &s.t.\ \forall i, j \in [1, n]\ and\ i \neq j. \end{aligned} \tag{12}$$

Plane-plane relationship. With the definition of distance D_{pl} between two planes, the plane layout consistency error is defined as:

$$\begin{aligned} L_{pl} = &\sum_{i,j} (D_{pl}(\pi_1^i, \pi_1^j) - D_{pl}(\pi_2^i, \pi_2^j)) \\ &s.t.\ \forall i, j \in [1, l]\ and\ i \neq j. \end{aligned} \tag{13}$$

To reduce the computation cost, we sample a subset of these layout relationships randomly when the number of feature matches is large.

5.3 Motion estimation

Based on the error terms analyzed in above, the feature-wise alignment error $E_{feature}$ in Equation (1) is defined as follows:

$$E_{feature} = w_{pt} \sum_{i=1}^{m} E_{pt}^i + w_{ln} \sum_{j=1}^{n} E_{ln}^j + w_{pl} \sum_{k=1}^{l} E_{pl}^k \tag{14}$$

where w_{pt}, w_{ln} and w_{pl} are the weights for points, lines and planes respectively. Specially, the weights w_{pt} and w_{ln} vary with the richness of the environment. In the low-texture environment, we often set w_{pt} small to reject the scarce point matches. And in the richly-textured environment, we decrease w_{ln} to reduce the impact of abundant inaccurate line matches. We set $w_{pl} = 0$ if no planes are detected.

We define the layout consistency error E_{layout} as:

$$E_{layout} = \alpha_{pt} L_{pt} + \alpha_{ln} L_{ln} + \alpha_{pl} L_{pl} \tag{15}$$

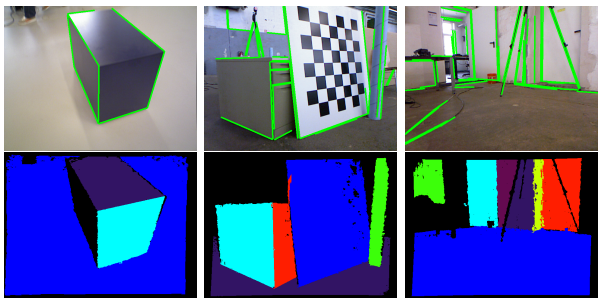


Figure 5: Sample images from TUM datasets with lines marked in green, and their plane detection results. From left to right: fr3/cabinet, fr3/pioneer-slam, and fr3/pioneer-slam3.

where α_{pt}, α_{ln} and α_{pl} are the weights of the layout consistency error between geometric primitives.

We model the optimization problem as a graph, and express the camera pose using Lie-algebra $se3$. For the total energy function $E = E_{feature} + E_{layout}$, we optimize it iteratively. At each iteration, we firstly fix the $E_{feature}$ and optimize E_{layout} to refine the primitives matches, rather than locations of primitives. Then we optimize $E_{feature}$ with E_{layout} fixed to estimate the camera motion using g2o [8]. To reduce the influence of incorrect matches, we also employ the RANSAC strategy, and randomly select a minimum subset (3 pairs of point and line matches) from whole point and line matches.

G2o optimizer is also used to solve the graph optimization problem. We use a similar strategy as in ORB-SLAM2 [11] to select new keyframes and add them into a global pose graph. Lastly, we perform the pose graph optimization to further optimize the whole camera poses.

6 EXPERIMENTAL RESULTS

We evaluate our method on the TUM RGB-D benchmark [16] which contains many RGB-D sequences with ground-truth camera trajectories. We use the Absolute Trajectory Error (ATE) to measure the difference between the estimated trajectory and the ground-truth. All experimental results are the median over 5 executions for each sequence. The sizes of color images and depth images are both 640×480. Fig. 5 gives some example images with extracted straight lines marked in green and corresponding plane detection results. It should be pointed out that we do not implement the loop closure module in our system, and we select some sequences that do not contain loops for evaluation and comparison.

6.1 Localization accuracy in the low-texture environment

To evaluate the performance of our system in the low-texture environment, we select some low-texture sequences from TUM datasets. In the first experiment, we compare our method with two typical RGB-D SLAM systems, point-based ORB-SLAM2 (ORB) [11] and direct-based Elastic Fusion (Elastic) [19]. The comparison results are summarized in Table 1.

Table 1: ATE (m) comparison with two typical RGB-D SLAM systems in the low-texture sequences

Sequence	Ours	ORB	Elastic
fr3/cabinet	**0.029**	X	0.947
fr3/large-cabinet	0.048	**0.041**	0.506
fr3/teddy	**0.014**	0.025	0.489
fr2/pionslam	**0.108**	X	1.983
fr2/pionslam2	**0.072**	X	1.459
fr2/pionslam3	**0.089**	X	1.583
fr3/str-notex-far	**0.017**	X	0.038
fr3/str-notex-near	**0.144**	X	0.535

* Results of ORB-SLAM2 and Elastic Fusion are executed with the open-source packages. X means that the tracking is lost and a large amount of frames are not processed by the system.

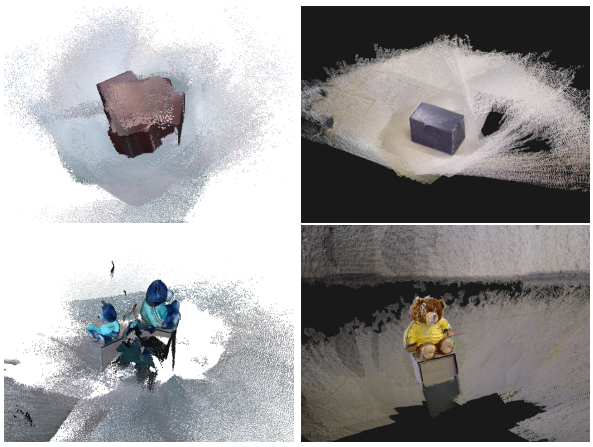


Figure 6: The reconstruction results obtained with Elastic Fusion (left) and our method (right).

There are totally 8 low-texture sequences, and our method outperforms two typical RGB-D SLAM systems in 7 sequences. Due to the lack of point features, ORB-SLAM2 fails in 6 sequences. The Elastic Fusion turns out not robust for large environment, especially in the three pionslam datasets.

As shown in Fig. 6, Elastic Fusion produces plenty of incorrect registrations, while our geometric primitives based method works well and builds a global consistent map. Direct-based methods usually rely on the dense matching, and converge easily to local minimum. Our method is based on the feature-based strategy and the layout consistency constraints, yielding the reliable matches and the optimal camera poses.

6.2 Localization accuracy in the richly-textured environment

We also evaluate our method in some richly-textured sequences, and compare it with some RGB-D SLAM systems, including RGB-D SLAM (RGB-D) [4], the PCL implementation of Kinect Fusion (KinFu) [12], Dense Visual SLAM (DVO) [7] and CPA-SLAM (CPA) [10].

Table 2: ATE (m) comparison in the richly-textured sequences

Sequence	Ours	RGB-D	KinFu	DVO	Elastic	ORB	CPA
fr1/xyz	0.019	0.014	0.026	0.011	0.011	**0.009**	0.011
fr1/rpy	0.026	0.026	0.133	**0.020**	0.034	0.021	0.024
fr1/desk	0.019	0.023	0.057	0.021	0.026	**0.016**	0.018
fr1/desk2	0.024	0.043	0.420	0.046	0.080	**0.022**	0.029
fr1/room	**0.042**	0.084	0.313	0.053	0.361	0.047	0.055
fr2/desk	0.012	0.057	0.034	0.017	0.264	**0.009**	0.046
fr3/str-tex-far	**0.009**	–	–	0.039	0.012	0.010	–
fr3/str-tex-near	0.011	–	–	0.041	0.016	**0.009**	–
average	**0.020**	0.041	0.164	0.031	0.100	**0.018**	0.031

* Results of RGB-D SLAM, KinFu and DVO are extracted from DVO paper. Results of CPA are from CPA-SLAM paper.

Table 3: ATE (m) comparison of each component in the low-texture sequences

Sequence	Point-only	Line-only	Plane-only	Point-Line	Point-Plane	Point-Line-Plane
fr2/pionslam	1.667	0.242	0.601	0.157	0.469	**0.108**
fr3/cabinet	0.530	0.029	0.635	0.035	0.316	**0.029**
fr3/str-notex-far	X	0.089	0.159	0.027	0.097	**0.017**
fr3/str-notex-near	X	0.166	0.154	0.166	0.154	**0.144**
average	1.099	0.132	0.387	0.096	0.259	**0.075**

The comparison results are shown in Table 2. The performance of ORB-SLAM2 is the best among these SLAM systems. The average trajectory error of ORB-SLAM2 in the 8 sequences is 0.018 m, and that of our method is 0.020 m. It should be noticed that our method performs better than CPA-SLAM which is based on the plane-model alignment.

In the richly-textured environment, our system provides comparable results with the state-of-the-art ORB-SLAM2, and outperforms other RGB-D systems in the average.

6.3 Ablation studies

In this section, we evaluate the function of each kind of geometric primitive in our method, including the point, line and plane. The layout consistency constraints of each primitive are also included when analyzing each component.

Table 3 shows the ATE comparison of each component in the low-texture environment. It is obvious that point-only based method always fails or produces heavy drift due to the lack of keypoints in the low-texture environment. Line-only based method can handle all of these sequences, and proves to be effective in the low-texture environment. Plane-only based method performs better than point-only method in 3 out of 4 sequences. However, it is not very robust when only using the plane features. In most cases, point-line fused method outperforms the single feature based methods. Point-plane fused method performs better than point-only or plane-only based methods. However, it is weaker than line-only based method.

Compared with the point-line fused method, point-line-plane fused method reduces the drift with an average improvement of 22%. The additional use of line features reduces the drift error with an average improvement of 71% compared with the point-plane fused method. Fig. 7 shows the effect of the plane alignment. It is clear that the plane alignment greatly eases the drift problem.

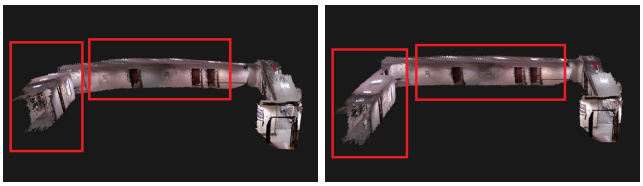


Figure 7: Comparison of the corridor reconstruction results obtained by point-line fused method without (left) / with (right) plane alignment. The plane alignment greatly reduces the drift in the red boxes.

Table 4: The computation time of each component in sequence f3/cabinet

Component	Time (ms)
Point detection + matching	2.08 + 0.54
Line detection + matching	22.17 + 0.02
Plane detection + matching	18.86 + 0.02
Motion estimation	28.08
Total	71.77

6.4 Runtime analysis

Finally, we show the computation time in Table 4. All experiments are measured on a PC with Intel Core i7 (8 cores @4.00 GHz) and 16GB memory. The runtime performance of our system highly depends on the richness of the environment texture. We choose a low-texture sequence f3/cabinet as an example which has about 50 point matches, 10 line matches

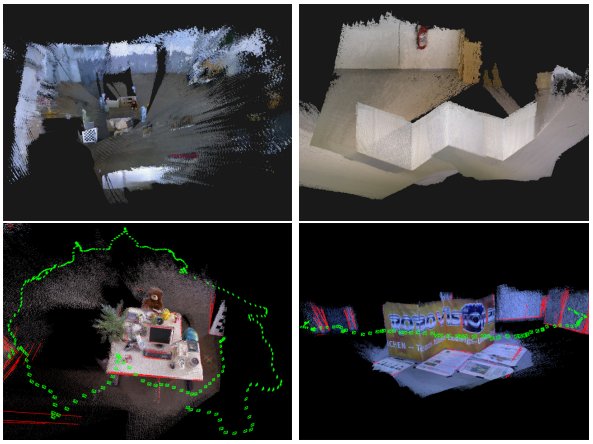


Figure 8: Dense point cloud reconstruction results.

and 3 plane matches in average. The plane detection and matching take about 19 ms and the remaining part takes about 53 ms. Our method can run at about 14 fps. To speed up the procedure, plane detection and matching are only performed in the keyframes and the speed can reach 20 fps.

7 CONCLUSIONS

In this paper, we propose the geometric primitives based RGB-D SLAM, a real-time feature-based system fusing point features, line features and plane features to recover camera motion in the low-texture environment. A unified solution is proposed to fuse these geometric primitives and the layout consistency constraints between them for camera tracking. Our method can handle the low-texture environment, and obtains comparable results with the state-of-the-art method in the richly-textured environment. Fig. 8 shows some dense point cloud models reconstructed with our method.

In the future, we will introduce the loop detection and closing techniques into our method to reduce the accumulated drift. Another direction is to fuse semantic analysis to build a high-level semantic map.

ACKNOWLEDGMENTS

The authors would like to thank the anonymous reviewers for their valuable comments and helpful suggestions. This work was supported by National Key Research and Development Program of China (No. 2016YFB1001501) and NSFC (No. 61379068, 61402387).

REFERENCES

[1] Cuneyt Akinlar and Cihan Topal. 2011. EDLines: A real-time line segment detector with a false detection control. *Pattern Recognition Letters* 32, 13 (2011), 1633–1642.

[2] Paul J Besl, Neil D McKay, et al. 1992. A method for registration of 3-D shapes. *IEEE Transactions on pattern analysis and machine intelligence* 14, 2 (1992), 239–256.

[3] JiaWang Bian, Wen-Yan Lin, Yasuyuki Matsushita, Sai-Kit Yeung, Tan-Dat Nguyen, and Ming-Ming Cheng. 2017. Gms: Grid-based motion statistics for fast, ultra-robust feature correspondence. In *2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. IEEE, 2828–2837.

[4] Felix Endres, Jurgen Hess, Jurgen Sturm, Daniel Cremers, and Wolfram Burgard. 2014. 3-D mapping with an RGB-D camera. *IEEE Transactions on Robotics* 30, 1 (2014), 177–187.

[5] Chen Feng, Y Taguchi, and V. R Kamat. 2014. Fast plane extraction in organized point clouds using agglomerative hierarchical clustering. In *IEEE International Conference on Robotics and Automation*. 6218–6225.

[6] Peter Henry, Michael Krainin, Evan Herbst, Xiaofeng Ren, and Dieter Fox. 2010. RGB-D mapping: Using depth cameras for dense 3D modeling of indoor environments. In *In the 12th International Symposium on Experimental Robotics (ISER)*. Citeseer.

[7] Christian Kerl, Jurgen Sturm, and Daniel Cremers. 2013. Dense visual SLAM for RGB-D cameras. In *Intelligent Robots and Systems (IROS), 2013 IEEE/RSJ International Conference on*. IEEE, 2100–2106.

[8] Rainer Kümmerle, Giorgio Grisetti, Hauke Strasdat, Kurt Konolige, and Wolfram Burgard. 2011. g2o: A general framework for graph optimization. In *Robotics and Automation (ICRA), 2011 IEEE International Conference on*. IEEE, 3607–3613.

[9] Yan Lu and Dezhen Song. 2015. Robust RGB-D Odometry Using Point and Line Features. In *IEEE International Conference on Computer Vision*. 3934–3942.

[10] Lingni Ma, Christian Kerl, Jörg Stückler, and Daniel Cremers. 2016. Cpa-slam: Consistent plane-model alignment for direct rgb-d slam. In *Robotics and Automation (ICRA), 2016 IEEE International Conference on*. IEEE, 1285–1291.

[11] Ral Mur-Artal and Juan D. Tards. 2017. ORB-SLAM2: An Open-Source SLAM System for Monocular, Stereo, and RGB-D Cameras. *IEEE Transactions on Robotics* 33, 5 (2017), 1255–1262.

[12] Richard A Newcombe, Shahram Izadi, Otmar Hilliges, David Molyneaux, David Kim, Andrew J Davison, Pushmeet Kohi, Jamie Shotton, Steve Hodges, and Andrew Fitzgibbon. 2011. KinectFusion: Real-time dense surface mapping and tracking. In *Mixed and augmented reality (ISMAR), 2011 10th IEEE international symposium on*. IEEE, 127–136.

[13] Albert Pumarola, Alexander Vakhitov, Antonio Agudo, Alberto Sanfeliu, and Francese Moreno-Noguer. 2017. PL-SLAM: Real-time monocular visual slam with points and lines. In *Robotics and Automation (ICRA), 2017 IEEE International Conference on*. IEEE, 4503–4508.

[14] Ethan Rublee, Vincent Rabaud, Kurt Konolige, and Gary Bradski. 2011. ORB: An efficient alternative to SIFT or SURF. In *IEEE International Conference on Computer Vision*. 2564–2571.

[15] Renato F Salas-Moreno, Ben Glocken, Paul HJ Kelly, and Andrew J Davison. 2014. Dense planar SLAM. In *Mixed and Augmented Reality (ISMAR), 2014 IEEE International Symposium on*. IEEE, 157–164.

[16] Jürgen Sturm, Nikolas Engelhard, Felix Endres, Wolfram Burgard, and Daniel Cremers. 2012. A benchmark for the evaluation of RGB-D SLAM systems. In *Intelligent Robots and Systems (IROS), 2012 IEEE/RSJ International Conference on*. IEEE, 573–580.

[17] Yuichi Taguchi, Yong-Dian Jian, Srikumar Ramalingam, and Chen Feng. 2013. Point-plane SLAM for hand-held 3D sensors. In *Robotics and Automation (ICRA), 2013 IEEE International Conference on*. IEEE, 5182–5189.

[18] Zhiheng Wang, Fuchao Wu, and Zhanyi Hu. 2009. MSLD: A robust descriptor for line matching. *Pattern Recognition* 42, 5 (2009), 941–953.

[19] T. Whelan, S. Leutenegger, R. F. Salas-Moreno, B. Glocker, and A. J. Davison. 2015. ElasticFusion: Dense SLAM Without A Pose Graph. In *Robotics: Science and Systems (RSS)*. Rome, Italy.

[20] Shichao Yang and Sebastian Scherer. 2017. Direct Monocular Odometry Using Points and Lines. In *Robotics and automation (ICRA), 2017 IEEE international conference on*. IEEE.

[21] Shichao Yang, Yu Song, Michael Kaess, and Sebastian Scherer. 2016. Pop-up SLAM: Semantic monocular plane SLAM for low-texture environments. In *Ieee/rsj International Conference on Intelligent Robots and Systems*. 1222–1229.

[22] Yizhong Zhang, Weiwei Xu, Yiying Tong, and Kun Zhou. 2015. Online structure analysis for real-time indoor scene reconstruction. *ACM Transactions on Graphics (TOG)* 34, 5 (2015), 159.

[23] Huizhong Zhou, Danping Zou, Ling Pei, Rendong Ying, Peilin Liu, and Wenxian Yu. 2015. Structslam: Visual slam with building structure lines. *IEEE Transactions on Vehicular Technology* 64, 4 (2015), 1364–1375.