REAL-TIME MONOCULAR VISUAL SLAM BY COMBINING POINTS AND LINES

Xinyu Wei, Jun Huang, Xiaoyuan Ma*

Shanghai Advanced Research Institute, Chinese Academy of Sciences, Shanghai, China
{weixy, huangj, maxy}@sari.ac.cn

ABSTRACT

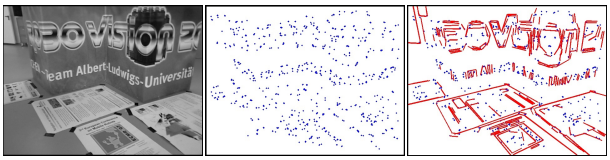
This paper presents a real-time monocular SLAM algorithm which combines points and line segments. We extend traditional point-based SLAM system with line features which are usually abundant in man-made scenes. The system is more robust and accurate than traditional point-based and direct-based monocular SLAM algorithms. In order to improve the timeliness of multi-feature based SLAM, we propose a novel feature level parallel processing framework and a fast line matching algorithm. For improving the reconstruction accuracy of 3D line segments which is usually affected by unreliable line endpoints, a sample point-based 3D reconstruction algorithm for line segments is proposed. Our system is implemented based on a popular monocular SLAM known as ORB-SLAM and tested on the TUM RGB-D benchmark. The experiment results demonstrate that the proposed system performs better than current state-of-the-art visual SLAM with respect to accuracy.

Index Terms— Visual SLAM, line segment, bundle adjustment, 3D reconstruction, line matching

1. INTRODUCTION

Simultaneous Localization and Mapping (SLAM) algorithm is used to estimate the map of the environment and the pose of a mobile robot simultaneously. Monocular SLAM, which uses monocular images as input and has the advantages of low price and compactness, is widely used in various fields such as augmented reality and autonomous vehicle navigation.

Existing monocular SLAM can be classified into two main categories with respect to the information which is used: feature-based methods [1, 2, 3, 4] and direct methods [5, 6, 7]. The feature-based methods use features extracted from images to estimate camera poses and a 3D map. In these methods, point is the most commonly used feature type. While many famous SLAM systems such as PTAM [1] and ORB-SLAM [2] are based on point features and have a good performance, the feature point based SLAM is still prone to fail in low-textured environments where it is difficult to find a reliable set of point features [8]. The primary cause of this problem is an important limitation that only information that conforms to the point feature can be used in feature point based SLAM system, which is also one major motivation for the latter methods [5]. The direct methods use the intensity values of all the image instead of features. These methods are able to exploit all the information in the image to build dense or semi-dense maps. But the benefits of robustness and invariance to photometric variations which is provided by features are sacrificed, and the camera localization accuracy of direct methods is generally lower than the current state-of-art feature point based methods such as ORB-SLAM [2].



(a) Source image (b) Point features (c) Points and lines

Fig. 1: Compared with point features extracted from image, combining points and line segments can provide more abundant information of the surroundings.

In order to tackle the problem in traditional feature point based SLAM systems, combination of points and line segments has been utilized in SLAM systems [9, 10, 11]. Compared with points, line segments can provide more geometrical structure information and are usually abundant in man-made scenes [12, 13] (see Fig. 1). Thus line segments are able to complement points well and improve the robustness and accuracy of feature point based SLAM system. Recently, Pumarola et al. proposed PL-SLAM [9], which is a monocular SLAM with points and lines. PL-SLAM improved the trajectory accuracy of point based SLAM system and could estimate an initial map using only lines. In [10], authors used the stereo camera to capture point and line features. Based on LSD [14] and LBD [15], they introduced an improved extraction and matching method for line features. In [11], a probabilistic stereo visual odometry system was proposed based on the combination of both point and line segment.

In a real-time SLAM system, processing line segments and points simultaneously is challenging. Firstly, high computational burden is required for the multi-feature processing tasks [8]. In the existing similar SLAM systems, only

*Corresponding author: Jun Huang (huangj@sari.ac.cn).

978-1-5386-9552-4/19/$31.00 ©2019 IEEE
DOI 10.1109/ICME.2019.00026

a few [9, 10] barely reached the real-time specifications (20 Hz). Secondly, compared with points, there exist more difficulties in the line segments matching and triangulation, which are primarily caused by the unreliable line endpoints. Focusing on these issues, we propose a feature level parallel processing framework, a fast line matching algorithm and a sample point based 3D reconstruction algorithm for line segments. The system is built upon the ORB-SALM, which is a famous feature point based monocular SLAM using ORB features [16]. The main contributions of the paper are as follows.

- We propose a real time monocular SLAM system using both points and line segments, and we conduct extensive experiments with comparison to current state-of-the-art visual SLAM using the TUM RGB-D dataset [17].
- We propose novel line processing algorithms in visual SLAM, which include a sample point based 3D reconstruction algorithm for line segments and a fast line matching algorithm.
- A feature level parallel processing framework is introduced. With this framework, the proposed multi-feature based system can run at around 30 Hz.

2. SYSTEM OVERVIEW

The proposed system is built upon ORB-SLAM [2]. Since the combined process of two kinds of features significantly increases the amount of calculation and complexity, the system is not a simple extension from ORB-SLAM. Novel line segment processing algorithms and a feature level parallel processing framework are adopted in the system (as shown in Fig.2). The main building blocks are briefly reviewed and the line segment related operations are presented in detail.

The system has three threads including tracking, local mapping and loop closing. The tracking localizes camera pose in every frame. The local mapping maintains the dynamic maps which includes keyframes, map points (3D points in map) and map lines (3D line segments in map). The loop closing constantly searches loop and corrects it. In Fig. 2, the feature level parallel processing modules are denoted with red doted boxes in which point and line segments are parallel processed. These modules are presented in detail in Section 4.1.

In our system, line segments are detected by the EDLines detector [18]. After feature detection, the initial pose is estimated with point matches. Line segments are not involved in this step because the existing line descriptors, i.e. MSLD [19] and LBD [15], do not perform as efficient as point descriptors, and that will reduce the efficiency of feature level parallelism. We only extract the LBD line descriptors from keyframes in local mapping thread. Once the initial estimation of the camera pose is completed, matches with the local map points and lines are searched, and camera pose is optimized with these

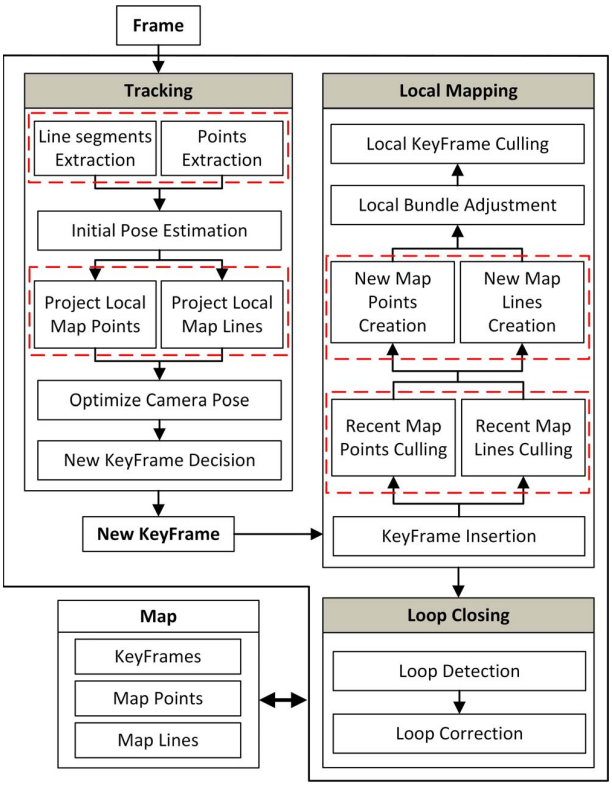


Fig. 2: Overview of the proposed SLAM system.

matches. The local map lines matching algorithm is explained in Section 3.1. When a keyframe is selected, the local mapping thread creates new map lines from it and culls the specific map lines according to an exigent policy. The new map lines creation is detailed in Section 3.2.

3. LINE FEATURES IN VISUAL SLAM

3.1. Line Segment Matching

In our system, there exist two line segment matching tasks: 2D-2D and 3D-2D line segment matching. The 2D-2D one is implemented based on the method in [15]. The 3D-2D line segment matching is conducted in three steps: projecting the 3D line segments in the image, determining candidate matches and identifying the best match.

The 3D line segment projection is essentially the projection of the endpoints. Due to occlusion and deformation of the segment, the detection of line endpoints is usually unreliable [20]. We seek to alleviate this issue by using observation angle to determine the most suitable endpoints. In our system, map line is represented as $L = (L_P, L_F)$. L_P is the 3D line containing the line segment which is parameterized with Plücker line coordinate and orthonormal representation [21]. L_F is a set of keyframes in which the line can be observed.

As illustrated in the left image in Fig. 3, we define the observation angle of L in the current frame F as the angle β between the principal axis L_z and the plane π_L containing L and the camera center C. The right image in Fig. 3 denotes the calculation process of the 3D endpoints. We first find the keyframe F_L which have the closest observation angle of L with current frame. Then the endpoint A of L are determined by the intersection of L and the plane π_a which contains camera center C and the vertical line l_a of ab. ab is the matching line segment of L in F_L. The other endpoint B is determined with the same method. At last, the endpoints A and B are projected in F and the projection line segment l_L of L can be obtained.

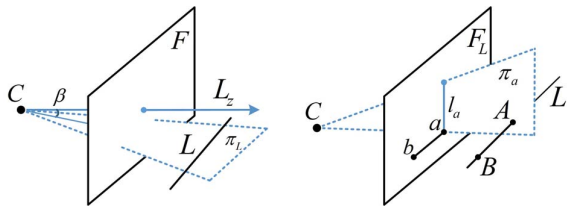


Fig. 3: Observation angle β of 3D line segment L (left). And the calculation of the endpoints A, B of L (right).

Candidate matching line segments of l_L in F is calculated based on the line direction. In our system, the line segments in current frame are divided into 36 sets according to their directions. And the point features are processed with the same method in ORB-SLAM, which divides the points into 64*48 sets based on their coordinates (see Fig. 4). The line segment groups in which the lines have the similar direction with l_L are selected. All line segments in these groups are added into the candidate matching line set S_L of L.

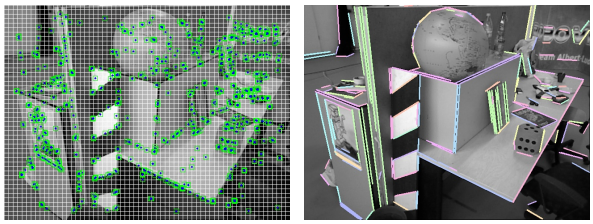


Fig. 4: Classification of the features. Green dots represent the point features and they are divided into 64*48 sets (left). Line segments of the same color indicate one set and all lines are classified into 36 groups (right).

Then the best matching line of L is calculated in S_L based on the nearest neighbor algorithm. Compared to point features, the matching error of line segments is hard to defined since it is multidimensional [20]. We comprehensively consider the endpoint-line distance and endpoint-endpoint distance. The matching error of segments l_L and l in our system is defined as:

$$d_1(l_L, l) = \alpha(d(l_L, p_1) + d(l_L, p_2)) \quad (1)$$

$$d_2(l_L, l) = (1-\alpha)(d'(p_1, a) + d'(p_2, b)) \quad (2)$$

$$d(l_L, l) = (d_1(l_L, l) + d_2(l_L, l))\frac{l_F^*}{l_L^*} \quad (3)$$

Where p_1, p_2 are the endpoints of l and a, b are the endpoints of l_L. $d(l_L, p_1)$ represents the distances of p_1 and l_L. $d'(p_1, a)$ is defined as the distance between p_1 and a. l_L^* is the length of l_L, and l_F^* is the average length of lines in current frame F. l_F^*/l_L^* can make the long line segments are more easily matched. α is a constant weight coefficients and the value is $0 \sim 1$. $d(l_L, l)$ is the final distance between l_L and l.

3.2. Map Line Initialization

The line initialization algorithms use line segment matches and camera poses to calculate new map lines. Due to possible noise in the data, accurate and robust map line initialization is a challenging task [21]. In order to solve this problem, we utilize sample points triangulation and linear fitting to create new map lines. The method contains two steps. First, 3D points are obtained according to sampling point matches. The second step is map lines fitting.

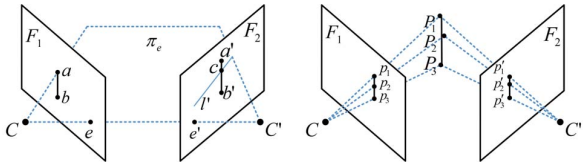


Fig. 5: Sampling points matching based on epipolar constraint (left) and map line fitting (right).

In order to get 3D sampling points, we first utilize the epipolar geometry of point to obtain accurate point matching pairs. The left image in Fig. 5 presents the calculation procedure of point matches. At first, n equidistant points are selected on line segments ab in frame F_1. For one point a, the epipolar line l' for a is calculated in the other frame F_2 [22]. Then the intersection point of l' and the matching line segment $a'b'$ in F_2 is the accuracy matching point of a.

Fig. 6 presents the accurate sampling point matches based on the epipolar geometry, the blue lines are the line segments extracted from images and the small purple circles in left image are sample points equidistantly selected on the line segments (here n is set to 5). The purple circles in right image are the intersection points of the matching lines and epipolar lines. Notice that the line segment extracted from the edge of paper in left image has different endpoints with the corresponding line segment in right image. But the correct point matches can still be obtained with our method.

Although above method is able to get accuracy point matches in many cases, there still exist a degeneracy when the

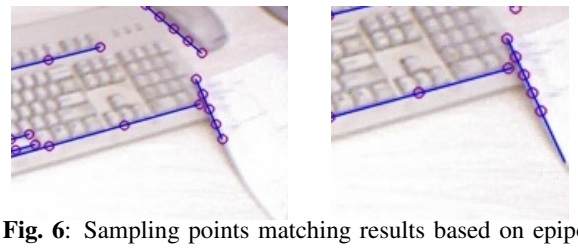


Fig. 6: Sampling points matching results based on epipolar geometry.

matching line $a'b'$ is parallel with epipolar lines l'. In order to settle this issue, sample points are selected on both matching lines and the length difference of two line segments is used to control the point matching. If the included angle of $a'b'$ and l' is greater than a threshold H_{th} and the length difference of them is less than L_{th}. Equidistant sampling points of ab and $a'b'$ are regarded as sampling point matches. After obtaining the point match (a, a'), the 3D point A can be calculated.

Finally, map line which has the best fit to the set of 3D points is obtained. The right image in Fig. 5 denotes the linear fitting process. (p_1, p_1') is a sampling point match and P_1 is the corresponding 3D point. The map line is calculated from (P_1, P_2, P_3). During this process, we can find the outliers in input data through checking the fitting error and the number of generated 3D points. The pseudocode of the map line initialization is given in Algorithm 1.

4. COMBINED TREATMENT OF POINTS AND LINES

4.1. Parallel Processing of Points and Lines

Parallel processing is an important method to build real-time visual SLAM. Since PTAM [1] splits tracking and mapping into two separate tasks and processes them in parallel threads, most subsequent visual SLAM systems adopt parallel processing algorithm and usually contain tracking, mapping and loop closing tasks [2, 9]. In our system, we extend the traditional parallel processing framework with the feature level parallel processing which further speeds up the multi-feature based visual SLAM.

The feature level parallel processing modules are presented with red doted boxes in Fig.2. In the tracking thread, line segments and points are extracted in two parallel threads. Considering the speed difference of line detector and point detector, the point detection and ORB descriptors extraction [16] are performed in one thread, and the line detection is performed in the other. After initial pose estimation, the map points projection and map lines projection are also performed in two threads. In the local mapping thread, map lines and map points are treated separately in features culling and new features creation which are desirable for parallel processing.

Algorithm 1 Map Line Initialization

Input: Line segments match pairs $(l_i, l_i^{'})(i = 0, 1, \ldots, n)$, camera projection matrices M, M', fundamental matrix F, number of sampling points sn.
Output: Map lines $L_i(i = 0, 1, \ldots, m)$.

```
for i = 0 → n do
    Obtain equidistant sampling point sets G_1 and G_2 based on l_i and l_i' respectively.
    Get the length difference L_d of l_i and l_i'.
    for j = 0 → sn do
        Compute the epipolar line l_e of the point G_1[j].
        Get the included angle h of l_e and l_i'.
        if h < H_th then
            Calculate point p of the intersection of l_e and l_i'.
            Generate 3D points X based on the 2D points p, G_1[j] and camera projection matrices M, M'.
        else if L_d < L_th
            Generate 3D points X based on the point G_2[j] and G_1[j].
        end if
        Add X into the set S. P_n indicates the size of S.
    end for
    if P_n > P_th then
        Conduct linear fitting and the fitting error is expressed as E.
        if E < E_th then
            Obtain the new map line L_i.
        end if
    end if
end for
```

4.2. Bundle Adjustment with Points and Lines

As stated before, the map in system is formed by the keyframes, map lines and map points. Bundle adjustment is adopted to optimize the poses of keyframes and the features in map. For that, we define T_k as the SE(3) pose of the k-th keyframe. P_i and L_j are the i-th map point and the j-th map line. The 3D-2D point match and line match are represented as (P_i, p_i) and (L_j, l_j). Then, the cost function C in the back-end optimization is:

$$d_{k,j} = \left\| \begin{matrix} l_j * \pi(\mathbf{K}, T_k, L_j^a) \\ l_j * \pi(\mathbf{K}, T_k, L_j^b) \end{matrix} \right\|^2 \tag{4}$$

$$d'_{k,i} = \|p_i - \pi(\mathbf{K}, T_k, P_i)\|^2 \tag{5}$$

$$C = \sum_{k,i,j} \rho(d_{k,i}^T \Omega_{k,i}^{-1} d_{k,i} + d_{k,j}^T \Omega_{k,j}^{-1} d_{k,j}) \tag{6}$$

Where $d_{k,j}$ and $d'_{k,i}$ refer to the line reprojection error [23] and point reprojection error. $\mathbf{K}$ is the camera calibration matrix. L_j^a and L_j^b are the 3D endpoints of L_j.

$\pi(\mathbf{K}, T_k, L_j^a)$ represents the point projection. $\Omega_{k,i}^{-1}$ and $\Omega_{k,j}^{-1}$ denote the inverse covariance matrices of points and lines, and ρ is the Huber robust cost function.

5. EXPERIMENTAL RESULTS

The proposed SLAM system is tested in the TUM RGB-D benchmark [17] which is widely used in monocular SLAM experiments. We compared our system with current state-of-the-art visual SLAM systems including ORB-SLAM [2], PL-SLAM [9], LSD-SLAM [5], PTAM [1] and RGBD-SLAM [24]. All experiments were implemented on an Intel Core i7-3770 (4 cores @3.4GHz) processor with 4 GB RAM. Because of the randomness in the systems, we conduct all experiments five times and report the median from all runs.

5.1. Localization accuracy

Fig.7 indicates the comparison of trajectories generated by our SLAM and ORB-SLAM. The results demonstrate that combining points and line segments is able to improve the accuracy of camera pose estimation. Table 1 shows the results generated from 11 sequences. The results of PTAM, LSD-SLAM and RGBD-SLAM are obtained from [2] and the results of PL-SLAM are obtained from [9] (we adopt the best results of the two systems proposed in [9]). We use the method in [2] to measure the absolute trajectory error. Our proposed SLAM is able to process all sequences and has better performance than ORB-SLAM in 9 sequences. Our algorithm also gets more accurate results than PL-SLAM in 7 sequences. In all 11 sequences, PL-SLAM and PTAM have 4 and 2 best results respectively. However, PTAM lost track in 5 sequences. Only in 5 sequences PL-SLAM achieves higher accuracy than ORB-SLAM. RGBD-SLAM lost track in 8 sequences and LSD-SLAM has higher error than other methods.

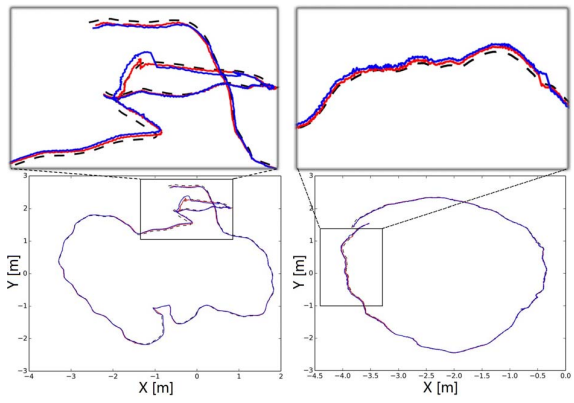


Fig. 7: Comparison of the trajectories obtained by ORB-SLAM (blue solid line) and the SLAM proposed (red solid line). The black dashed line represents the ground truth.

Table 1: Localization accuracy in the TUM RGB-D dataset.

TUM RGB-D Sequence	Absolute KeyFrame Trajectory RMSE (cm)					
	Ours	PL-SLAM	ORB-SLAM	PTAM	LSD-SLAM	RGBD-SLAM
f1_xy	**0.87**	1.21	0.94	1.15	9	1.34
f2_xyz	0.25	0.43	0.23	**0.2**	2.15	1.42
f3_long_office	**1.10**	1.97	1.68	-	38.53	-
f3_nstr_tex_near	**1.40**	1.58	1.43	2.74	7.54	-
f3_str_tex_far	0.97	**0.89**	1.05	0.93	7.95	-
f3_str_tex_near	1.16	1.25	1.19	**1.04**	-	-
f2_desk_person	**0.64**	1.99	0.72	-	31.73	2
f3_sit_xyz	0.81	**0.066**	0.89	0.83	7.73	-
f3_sit_halfsph	1.56	**1.31**	1.40	-	5.87	-
f3_walk_xyz	**1.17**	1.54	1.56	-	12.44	-
f3_walk_halfsph	1.68	**1.6**	1.93	-	-	-

5.2. Computation Time

The computation burden of multi-feature based Visual SLAM is the main problem need to be settled. We test the running time of our SLAM system in TUM RGB-D benchmark. The time required for each task in the tracking and local mapping threads presented in Table 2. Our system is compared with PL-SLAM and ORB-SLAM. In addition, we also compare the implementation of our SLAM system which utilizes the feature level parallel processing framework (Ours-A) and the implementation without the framework (Ours-B). The results of PL-SLAM are obtained from [9].

Table 2: Running time of each operation (ms).

Thread	Operation	Mean execution time (ms)			
		Ours-A	Ours-B	PL-SLAM	ORB-SLAM
Local Mapping	KeyFrame Insertion	15.38	19.18	17.08	13.99
	Map Feature Culling	0.44	0.49	1.18	0.08
	Map Feature Creation	190.94	231.22	74.64	76.74
	Local BA	154.06	179.68	218.25	107.95
	KeyFrame Culling	2.94	3.78	12.7	6.27
	Total	363.76	434.35	323.85	205.03
Tracking	Features extraction	14.94	24.37	31.32	12.36
	Initial Pose Estimation	4.82	4.75	7.16	3.27
	Track Local Map	9.22	9.41	12.58	6.93
	Total	28.98	38.53	51.06	22.56

Since adding line features to the visual SLAM increases the computational complexity. Our system and PL-SLAM are slower than ORB-SLAM. But Compared with PL-SLAM,

both implementations of our system have faster tracking speed than PL-SLAM (even if the experimental platform of PL-SLAM has better performance than ours), and our system with the feature level parallel processing framework reaches the real-time operation (30 Hz).

6. CONCLUSION

In this paper, we propose a real-time monocular SLAM system using point features and line features based on ORB-SLAM [2]. With the abundant structure information of scenes provided by line segments, the system is more robust and accuracy than traditional point-based SLAM. Multi-feature based SLAM systems are usually more accuracy than point-based SLAM, but the cost is slow running speed. In our system, the potential of feature level parallel processing in multi-feature based SLAM system is realized to improve the running speed. With the real time problem solved, SLAM system with both points and lines could get accuracy result and real-time performance at the same time. We have also presented a novel line initialization algorithm and a line matching algorithm to fix the problems of line processing in visual SLAM. We validate the system in TUM GRB-D benchmark, confirming the robustness and the real-time performance of the proposed system.

7. REFERENCES

[1] Georg Klein and David Murray, "Parallel tracking and mapping for small ar workspaces," in *ISMAR*. IEEE, 2007, pp. 225–234.

[2] Raul Mur-Artal, Jose Maria Martinez Montiel, and Juan D Tardos, "Orb-slam: a versatile and accurate monocular slam system," *IEEE Transactions on Robotics*, vol. 31, no. 5, pp. 1147–1163, 2015.

[3] Lilian Zhang and Reinhard Koch, "Hand-held monocular slam based on line segments," in *Irish Machine Vision and Image Processing Conference*. IEEE, 2011, pp. 7–14.

[4] Jakob Engel, Vladlen Koltun, and Daniel Cremers, "Direct sparse odometry," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 40, no. 3, pp. 611–625, 2018.

[5] Jakob Engel, Thomas Schöps, and Daniel Cremers, "Lsd-slam: Large-scale direct monocular slam," in *ECCV*. Springer, 2014, pp. 834–849.

[6] Richard A Newcombe, Steven J Lovegrove, and Andrew J Davison, "Dtam: Dense tracking and mapping in real-time," in *ICCV*. IEEE, 2011, pp. 2320–2327.

[7] Matia Pizzoli, Christian Forster, and Davide Scaramuzza, "Remode: Probabilistic, monocular dense reconstruction in real time," in *ICRA*. IEEE, 2014, pp. 2609–2616.

[8] Ruben Gomez-Ojeda, Jesus Briales, and Javier Gonzalez-Jimenez, "Pl-svo: Semi-direct monocular visual odometry by combining points and line segments," in *IROS*. IEEE, 2016, pp. 4211–4216.

[9] Albert Pumarola, Alexander Vakhitov, Antonio Agudo, Alberto Sanfeliu, and Francese Moreno-Noguer, "Pl-slam: Real-time monocular visual slam with points and lines," in *ICRA*. IEEE, 2017, pp. 4503–4508.

[10] Xingxing Zuo, Xiaojia Xie, Yong Liu, and Guoquan Huang, "Robust visual slam with point and line features," in *IROS*. IEEE, 2017, pp. 1775–1782.

[11] Ruben Gomez-Ojeda and Javier Gonzalez-Jimenez, "Robust stereo visual odometry through a probabilistic combination of points and line segments," in *ICRA*. IEEE, 2016, pp. 2521–2526.

[12] Guoxuan Zhang and Il Hong Suh, "Sof-slam: Segments-on-floor-based monocular slam," in *IROS*. IEEE, 2010, pp. 2083–2088.

[13] Guoxuan Zhang and Il Hong Suh, "Building a partial 3d line-based map using a monocular slam," in *ICRA*. IEEE, 2011, pp. 1497–1502.

[14] Rafael Grompone Von Gioi, Jeremie Jakubowicz, Jean-Michel Morel, and Gregory Randall, "Lsd: A fast line segment detector with a false detection control," *Transactions on Pattern Analysis and Machine Intelligence*, vol. 32, no. 4, pp. 722–732, 2010.

[15] Lilian Zhang and Reinhard Koch, "An efficient and robust line segment matching approach based on lbd descriptor and pairwise geometric consistency," *Journal of Visual Communication and Image Representation*, vol. 24, no. 7, pp. 794–805, 2013.

[16] Ethan Rublee, Vincent Rabaud, Kurt Konolige, and Gary Bradski, "Orb: An efficient alternative to sift or surf," in *ICCV*. IEEE, 2011, pp. 2564–2571.

[17] Jürgen Sturm, Nikolas Engelhard, Felix Endres, Wolfram Burgard, and Daniel Cremers, "A benchmark for the evaluation of rgb-d slam systems," in *IROS*. IEEE, 2012, pp. 573–580.

[18] Cuneyt Akinlar and Cihan Topal, "Edlines: A real-time line segment detector with a false detection control," *Pattern Recognition Letters*, vol. 32, no. 13, pp. 1633–1642, 2011.

[19] Zhiheng Wang, Fuchao Wu, and Zhanyi Hu, "Msld: A robust descriptor for line matching," *Pattern Recognition*, vol. 42, no. 5, pp. 941–953, 2009.

[20] Jonas Witt and Uwe Weltin, "Robust stereo visual odometry using iterative closest multiple lines," in *IROS*. IEEE, 2013, pp. 4164–4171.

[21] Adrien Bartoli and Peter Sturm, "Structure-from-motion using lines: Representation, triangulation, and bundle adjustment," *Computer vision and image understanding*, vol. 100, no. 3, pp. 416–441, 2005.

[22] Richard Hartley and Andrew Zisserman, *Multiple view geometry in computer vision*, Cambridge university press, 2003.

[23] Adrien Bartoli and Peter Sturm, "The 3d line motion matrix and alignment of line reconstructions," in *CVPR*. IEEE, 2001.

[24] Felix Endres, Jürgen Hess, Jürgen Sturm, Daniel Cremers, and Wolfram Burgard, "3-d mapping with an rgb-d camera," *IEEE Transactions on Robotics*, vol. 30, no. 1, pp. 177–187, 2014.