Direct Line Guidance Odometry

Shi-Jie Li[1], Bo Ren[1], Yun Liu[1], Ming-Ming Cheng[1], Duncan Frost[2], Victor Adrian Prisacariu[2]

***Abstract*— Modern visual odometry algorithms utilize sparse point-based features for tracking due to their low computational cost. Current state-of-the-art methods are split between indirect methods that process features extracted from the image, and indirect methods that deal directly on pixel intensities. In recent years, line-based features have been used in SLAM and have shown an increase in performance albeit with an increase in computational cost. In this paper, we propose an extension to a point-based direct monocular visual odometry method. Here we that uses lines to guide keypoint selection rather than acting as features. Points on a line are treated as stronger keypoints than those in other parts of the image, steering point-selection away from less distinctive points and thereby increasing efficiency. By combining intensity and geometry information from a set of points on a line, accuracy may also be increased.**

I. INTRODUCTION

Visual Odometry (VO) [1] and *Simultaneous Localization and Mapping (SLAM)* [2] are long studied topics in the field of robots, made popular today due to new applications in augmented reality, self-driving cars, and unmanned aerial vehicles. Visual odometry and SLAM are used with a number of different sensor types such as monocular cameras [3]–[5], stereo cameras [6], or RGB-D cameras [7]. Among these sensors, the monocular camera is particularly popular due to its low cost, power consumption, and weight.

Current SLAM approaches can be broadly divided into two categories, namely *direct* and *indirect* methods. Indirect methods [8], [9] extract keypoints from the image and rely on strong feature matching to solve data association. A sparse map for localization is constructed and refined by minimizing geometric error between landmark reprojections and keypoint measurements. These methods are able to provide tracking with high precision if enough keypoints are available, however they easily fail in featureless environments where insufficient keypoints are extracted from the image. In contrast, the direct methods [10], [11] skip the procedure of feature extraction and directly operate on the raw pixel intensities using photometric error models. As camera localization doesn't require point extraction, they fair better in featureless environments. For mapping, point extraction generally uses a weak descriptor (*i.e.* gradient), allowing enough points to be extracted in these cases. The performance of direct method is sensitive to the environment changes (such as luminance) which is not desired.

In an attempt to balance tracking accuracy with robustness line-based methods are proposed recently. [12] [13] [5] [4] Compared with the point-based features, lines are less sensitive to luminance changes. For example, in a homogeneous environment, there not existing many keypoints due to small image gradient, but we can usually detect many edges in it. Line-based methods are particularly useful in many man-made environments where discriminative keypoints are quite rare. Current point-line-based methods [4] [5] usually treat the line as a stronger feature and treat it similarly to keypoints. This method is usually computationally costly, negatively affecting the runtime performance of the SLAM system.

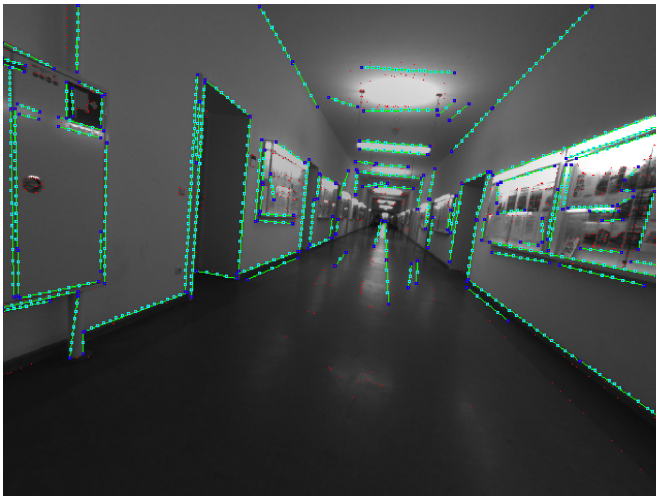


Fig. 1. A keyframe in DLGO. We extract points and lines on the keyframe. The positions of the extracted points and lines are updated using the incoming frame. The red points represent the Immature Point (IP) and the green lines denote the Immature Line (IL). On each line, the dark blue points represent the Immature Line End Point (ILEP) and the light blue points represent the ILCP (Immature Line Common Point).

In this paper, we propose a direct point-line-based visual odometry called *direct line guidance odometry* (DLGO, Fig. 1). The method is an extension to a popular direct point-based method [10]. In the traditional direct point-based methods, extracted points are treated independently, ignoring possible relationships between them. For our proposed method, we leverage the fact points on a line in 3D space remain collinear from frame to frame. For this reason we selectively use points that lie on lines in the image. Points on each line are made more distinctive by combining their intensity and geometry information together without increasing computational cost. By removing points that don't lie on lines, we further decrease computational cost. After each tracking and mapping phase we adjust the positions of points belonging to the same line, assuming points belonging to the same line collinear throughout their

[1] College of Computer and Control Engineering, Nankai University Tianjin, China;
[2] Department of Engineering, Oxford University, UK

lifetime. In doing so, geometry information from the line is fused into the system, thereby increasing accuracy. Our point-line extension outperforms or gets similar performance in terms of accuracy and robustness compared to state-of-the-art point-based visual odometry methods without sacrificing speed.

In the following sections, we first give a brief introduction of some recent vSLAM methods using points and lines in Sec. II. Then we give an overview of our system in Sec. III. In Sec. IV, we describe the tracking part in detail. The mapping procedure will be introduced in Sec. V. In Sec. VI, we provide the experimental comparison with some state-of-the-art algorithms.

II. RELATED WORK

In this section, we briefly introduce some recent vSLAM and VO methods.

1) Indirect Point-Based methods [9] [8] [14] [15]: First, the keypoints are extracted in the current frame. Then the descriptor for each point is computed (such as ORB [16], SURF [17]). The keypoints are matched by the difference between their descriptors. Next, a sparse 3D map is constructed by these matches. Finally, the pose is computed by PnP [18] or frame-to-frame tracking. These methods are easily lost for tracking in the featureless environments because of the insufficient keypoints.

2) Direct Point-Based methods [3] [10] [19] [20] [11] [21]: Compared with the former one, Direct Point-Based methods skip the procedure of feature extraction and feature matching. It directly utilizes the intensity value of pixels and computes the camera pose by minimizing photometric error. According to the size of used points, we can divide these methods into sparse methods [10] that only use sparse points in the image, semi-dense methods [19] [11] that use part of points in the image and dense methods [20] that use all points in the image. Although these methods are very robust, it is more likely influenced by some environment changes.

3) Indirect Point-Line-Based methods [4]: In these methods, the points are viewed in the same way as in the indirect point-based methods. The lines are usually treated similarly to points. The lines are first detected in the image. Then the descriptor of each line is computed. Next, the lines are matched according to the difference of their descriptors. Finally, both the lines and points are used in tracking and mapping. Although these methods can increase the robustness of the indirect point-based methods, the cost of line related computation is high.

4) Direct Point-Line-Based methods [5]: These methods usually treat the lines in the similar way as in the indirect point-line-based methods. Since the accuracy of the direct point-based methods is unsatisfactory, these methods use the line based features to improve the accuracy of tracking. However, the computation load related to the line for these methods is also too high, which means the efficiency is low.

III. METHOD

In this section, we give a brief overview of the proposed method.

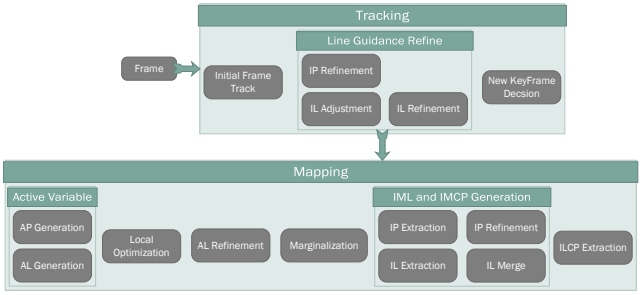


Fig. 2. System Overview. The method is split into tracking and mapping threads. In tracking thread, the incoming frame is tracked relative to the last keyframe. After that, the positions of IPs and ILs are refined. The mapping part is used to create the new keyframe and optimizes all APs and ALs on the active keyframes. The line information is used as a guidance both in tracking and mapping. The definitions of IPs, ILs, APs and ALs are shown in Fig. 3.

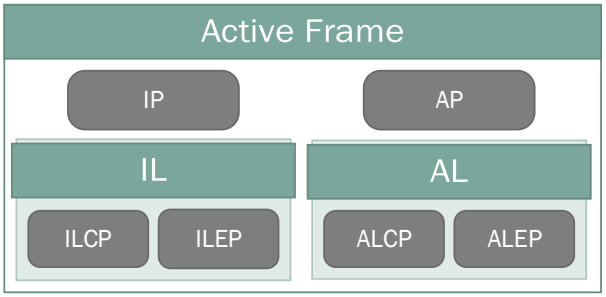


Fig. 3. Point Management. The points and lines are separated to immature part and active part. The immature part is not considered in the optimization. On the contrary, the active part takes part in the optimization. We use the following name to denote different type of point and line. Immature part: IP - Immature Point, IL - Immature Line, ILEP - Immature Line EndPoint, ILCP - Immature Line Common Point. Active elements: AP - Active Point, AL - Active Line, ALEP - Active Line EndPoint, ALCP - Active Line Common Point.

A. System pipeline

As our method is an extension to [10], the structure of our method follows a similar pattern. This is shown in Fig. 2.

Due to the dependence on the point gradient, the direct methods are usually affected by luminance changes seriously. To reduce this effect, our method follows the pipeline of [10] which takes into account the luminance changes and adopts a more complex calibration model to reduce this effect [22].

The whole system is constituted by tracking and mapping. In tracking thread, the incoming frame is tracked to the newest keyframe. After that, the positions of IPs and ILs are refined. (IPs and ILs are defined in the next subsection.) The details of this part are shown in Sec. IV. In Mapping thread, a new keyframe is created first. Then a local optimization is conducted using a sliding window filter for all variables in active keyframes. Finally, the new IPs and ILs are extracted on the new keyframe. We give a detailed description of this part in Sec. V.

B. Point-Line Management

In this work, we adopt the common inverse depth representation as the point parameterization [23]. A point is extracted and described by the gradients in its neighbors. To improve the system efficiency, we employ the points sampled on the line into the system instead of using a complicated line descriptor. The line is described by the points sampled on it. To manage the points on a line, we denote a line by its two endpoints. According to whether or not being considered in the optimization, the points and lines can be divided into two parts. The immature part is not considered in the optimization. On the contrary, the active part takes part in the optimization. For the immature part, point is called IP (Immature Point) and line is denoted as IL (Immature Line). The two endpoints of IL are called ILEP (Immature Line End Point) and other points on IL are denoted as ILCP (Immature Line Common Point). We use similar name in active part, such as AP (Active Point), AL (Active Line), ALEP (Active Line End Point) and ALCP (Active Line Common Point). The relationship of these variables is shown in Fig. 3.

C. Photometric Error Model

The photometric error is defined in a similar way as [10]. The photometric error of a point $\boldsymbol{p} \in \Omega_i$ in reference frame I_i observed in a target frame I_j is defined as :

$$E_{\boldsymbol{p}j} := \sum_{\boldsymbol{p} \in \mathcal{N}_{\boldsymbol{p}}} w_{\boldsymbol{p}} ||(I_j[\boldsymbol{p'}] - b_j) - \frac{t_j e^{a_j}}{t_i e^{a_i}}(I_i[\boldsymbol{p}] - b_i)||_\gamma, \quad (1)$$

where $\mathcal{N}_{\boldsymbol{p}}$ is the set of pixels in the pattern, t_i and t_j are exposure times of the images I_i, I_j and $||*||_\gamma$ is the Huber norm. The $w_{\boldsymbol{p}}$ is a gradient-dependent weight:

$$w_{\boldsymbol{p}} := \frac{c^2}{c^2 + ||\nabla I_i(\boldsymbol{p})||_2^2}. \quad (2)$$

c is a fixed scalar and $\boldsymbol{p'}$ represents the projected point of $\boldsymbol{p}$ with inverse depth $d_{\boldsymbol{p}}$:

$$\boldsymbol{p'} = \pi_{\boldsymbol{C}}(\boldsymbol{R}\pi_{\boldsymbol{C}}^{-1}(\boldsymbol{p}, d_{\boldsymbol{p}}) + \boldsymbol{t}). \quad (3)$$

where $\boldsymbol{R}$ and $\boldsymbol{t}$ represent the rotation and translation between the images I_i and I_j, respectively. $\boldsymbol{C}$ denotes the intrinsic camera parameters. $\pi_C : \mathbb{R}^3 \to \Omega$ is the projection from camera coordinate to the image plane using intrinsic parameters $\boldsymbol{C}$ and $\pi_C^{-1} : \Omega \times \mathbb{R} \to \mathbb{R}^3$ is the inverse procedure. The a_i, a_j, b_i, b_j are photometric parameters in [22].

The line photometric error of a line $\boldsymbol{l} \in \mathcal{L}$ is the sum of point errors on it:

$$E_{\boldsymbol{l}j} := \sum_{e \in \mathcal{E}} E_{\boldsymbol{e}j} + \sum_{p \in \mathcal{P}} E_{\boldsymbol{p}j}, \quad (4)$$

where e represents the endpoints $\mathcal{E}$ and p represents all other sampled points $\mathcal{P}$ on the line.

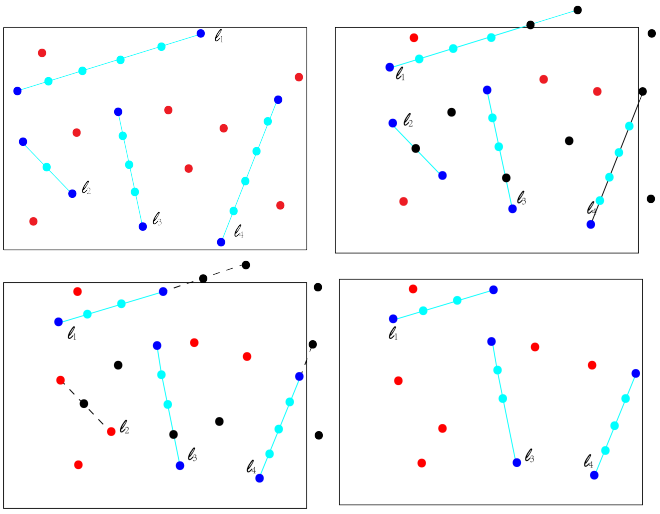


Fig. 4. IL Adjustment. The red point denotes IP; the light blue line denotes IL. On the IL, the dark blue point denotes ILEP and the light blue point denotes ILCP. The black point represents the point that fails to track in the current frame. The top left image shows the IPs and ILs in an active keyframe. The up right image shows the matches to the IPs and ILs in the active keyframe. In the down left image, we show the process of IL adjustment. The l_1 removes an ILEP and an ILCP. The l_2 removes an ILCP and only two ILEPs are left, so this line will be released and two ILEPs on it turn into IPs. The l_3 only remove an ILCP so the length of this line is not changed.The l_4 remove an ILEP only. The down right image shows the final result of line adjustment.

IV. TRACKING

This section introduces the tracking part of the whole system. In tracking thread, the incoming frame is first tracked to the latest keyframe. Then the positions of IPs and ILs in all active keyframes are refined using the matching on incoming frame. Finally, if the incoming frame is far from the closest active keyframe, it will go into the mapping thread.

A. Initial Frame Tracking

The incoming frame is tracked to the latest key frame using conventional two-frame direct image alignment, a multi-scale image pyramid and a constant motion model to initialize [10]. If the direct image alignment fails, the system conducts a recovery-tracking on the coarsest pyramid level by initializing with up to 27 different small rotations in different directions.

B. Line Guidance Refinement

In this process, the positions of IPs and ILs in all active keyframes are refined.

1) IP Refinement: The IP in an active keyframe is searched in the current frame using a discrete search along the epipolar line by minimizing the photometric error defined in equation (2). The matched point in the current frame is used to update the depth of corresponding IP in the active keyframe. If the match point to an IL is out of the view or the uncertainty of the updated depth is too large, this IP is marked as a refined failure point.

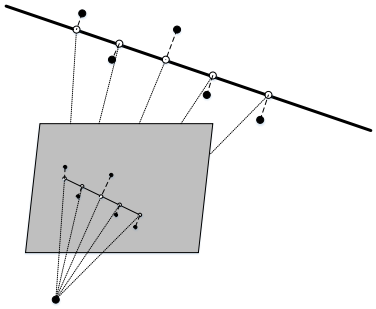


Fig. 5. IL refinement. In this process, The points on IL are first unprojected into 3D space. A 3D line is then created by these points. Next, this line is reprojected onto the image as the new IL. Finally, the new ILEP and ILCP are accessed by projecting the old one on the new IL.

2) IL Adjustment: For an IL, the ILCPs and ILEPs on it are first refined in the same way as IP. If an ILCP refines fail, we directly remove it from the IL. As for a refined failure ILEP, the system checks all refinement status of ILEPs and ILCPs on the same IL and removes the refined failure points. The new IL is composed of the remaining points. When an IL contains less than three points, this IL will be released and the remaining points on it turn into the IPs. This is because any two points are collinear. The whole IL adjust procedure shown in Fig. 4.

C. IL Refinement

After the IL adjustment, the positions of points on a refined success IL are updated by combining the line prior. This procedure can be represented as the update to the probability of points position on a line:

$$p(\boldsymbol{p}_p, \boldsymbol{p}_l) = p(\boldsymbol{p}_p|\boldsymbol{p}_l) \cdot p(\boldsymbol{p}_l) \tag{5}$$

where $\boldsymbol{p}_p$ denotes the point position on a IL and $\boldsymbol{p}_l$ represents the corresponding IL position.

The refined success ILs are first unprojected into the 3D space by unprojecting all ILEPs and ILCPs on it to the 3D space. Then a new 3D line is fitted according to these projected points. Next, this newly generated line is reprojected onto the image plane as the new IL. Finally, the new ILEPs and ILCPs are accessed by projecting the old one on the new IL.

Different from the traditional direct methods that treat points independent, we take into account the relationship of points on the same line. Here, we use the IL information to guide the positions of ILEPs and ILCPs on it. By keeping the points on the same line collinear in their lifetime, the ILCPs and ILEPs combine both the gradient and geometry information. With more information, the accuracy of the system is improved. The whole IL refinement procedure is shown in Fig. 5

D. New KeyFrame Decision

In the last step of the tracking thread, the system creates a new keyframe. The creation of a new keyframe depends on the field of view changes, camera translations and camera exposure time changes. A weighted sum considering these factors is used to create a new keyframe.

V. MAPPING

In mapping thread, a new keyframe is first created by projecting the APs and ALs in all active keyframes into it. The projected APs and ALs are slightly dilated, creating a semi-dense depth map. The positions of APs and ALs, camera poses and camera parameters are optimized in a sliding window filter together later. During the optimization, AL Refinement is executed after each iteration. For efficiency consideration, we will marginalize the APs, ALs and active frames that far from the current frame. Finally, the IPs and ILs on the new keyframe are generated.

A. Sliding Window Optimization

In this work, we use the sliding window optimization method. Before the optimization, the ILs and IPs in all keyframes are first activated as APs and ALs. The positions of ILs and IPs are treated as the initial value in APs and ALs. Then the APs and ALs take part into the optimization. In optimization, ALEPs and ALCPs are processed in the same way as APs.

The full photometric error over all frames, points and lines is given by:

$$E_{photo} := E_{\boldsymbol{p}} + E_{\boldsymbol{l}} \tag{6}$$

where the $E_{\boldsymbol{p}}$ represents the total error on APs and $E_{\boldsymbol{l}}$ denotes the total error on ALs.

$$E_{\boldsymbol{p}} := \sum_{i \in \mathcal{F}} \sum_{\boldsymbol{p} \in \mathcal{P}_i} \sum_{j \in obs(\boldsymbol{p})} E_{\boldsymbol{p}j} \tag{7}$$

$$E_{\boldsymbol{l}} := \sum_{i \in \mathcal{F}} \sum_{\boldsymbol{l} \in \mathcal{L}_i} \sum_{j \in obs(\boldsymbol{l})} E_{\boldsymbol{l}j} \tag{8}$$

where i runs over all frames $\mathcal{F}$, $\boldsymbol{p}$ over all APs $\mathcal{P}_i$ in frame i, and j over all frames $obs(\boldsymbol{p})$ in which the AP $\boldsymbol{p}$ is visible. $\boldsymbol{l}$ over all ALs $\mathcal{L}_i$ in frame i, and j over all frames $obs(\boldsymbol{l})$ in which the AL $\boldsymbol{l}$ is visible. The point error $E_{\boldsymbol{p}}j$ and the line error $E_{\boldsymbol{l}}j$ are defined in (2) and (5).

To keep more ALs in the current frame, we adopt a larger window size of active keyframes.

The total error (7) is optimized by the Gauss-Newton algorithm. The more mathematical detail can be found in [24]

B. AL Refinement

After each iteration in optimization, an AL Refinement procedure is applied on AL. This procedure is similar to the IL refinement in Sec. IV. In the Gauss-Newton algorithm, the solution usually treats each variable independently. However, in our situation, the ALCPs and ALEPs are collinear. And this characteristic will be changed in optimization procedure, which is not desired. Hence after each iteration, the AL Refinement is carried out to keep this characteristic. Finally, the APs, ALs and active frames that far from the current frame will be marginalized by Schur complement [24]. This

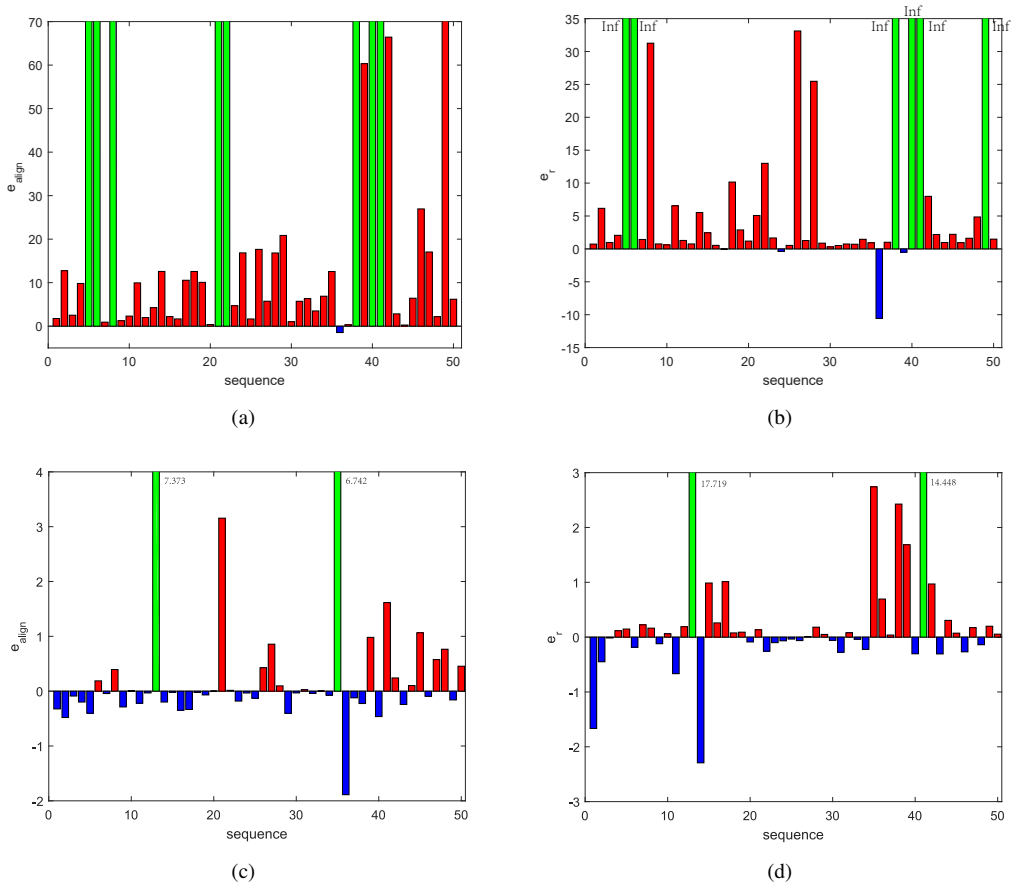
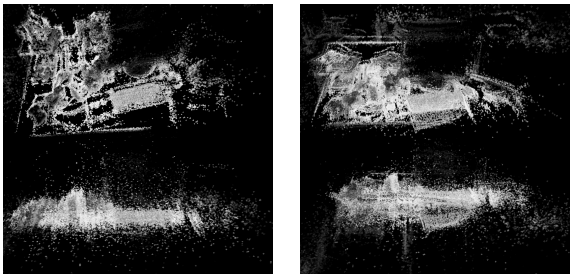


Fig. 6. The results are denoted by the difference between ORB-SLAM and DSO with our methods on alignment error e_{align} and accumulated rotational drift e_r. (a) shows e_{align} of ORB-SLAM vs Ours, (b) shows e_r of ORB-SLAM vs Ours, (c) shows e_{align} of ORB-SLAM vs Ours, (d) shows e_r of DSO vs Ours. The red color denotes our method outperforms ORB-SLAM/DSO whereas the blue color denotes ORB-SLAM/DSO outperforms our method. The green color represents the error beyond the limits. We use the value on them to show the error value. From these figures, we can see that our method outperforms ORB-SLAM and DSO in both alignment error and accumulated rotational drift in general.



(a) DLGO (b) DSO

Fig. 7. An example on a sequence belonging to TUM RGBD Dataset. The left column is the result of our method and the right column is produced by DSO. From the image we can see that our method achieves smaller drift than DSO.

operation can keep the number of variable in optimization in a fixed range. Hence the computation efficiency will not decrease as the size of frames increasing.

C. IL and IP Generation

In this procedure, ILs and IPs are created on the new keyframe. The main reason for a poor tracking accuracy of the direct methods is that direct methods adopt a less distinctive descriptor (*i.e.* gradient). In our approach we extract the points from the line structures. One advantage of this is that the extracted points have high gradients, while containing line-structural information at the same time. In the direct methods, there are too many points concentrating near the line. Hence, we reduce the computational cost by selecting a subset of points from the lines.

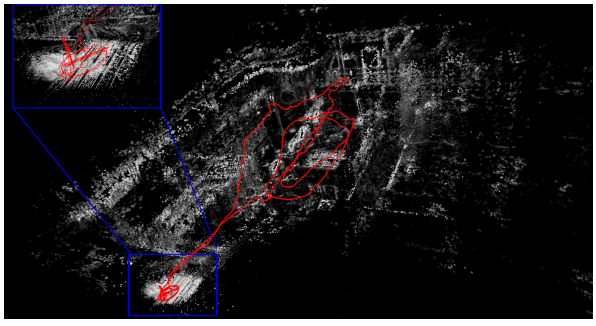


Fig. 8. This figure shows the final map on sequence Machine Hall 02 in EuRoc MAV Dataset

1) IP Generation: For the IP generation, a 3-layer image pyramid is constructed. Then on each layer, the image is

	Absolute trajectory error (m)			Relative pose error (m)		
sequence	Ours	DSO	ORB-SLAM	Ours	DSO	ORB-SLAM
fr1_xyz	**0.0538**	0.069	0.0836	**0.0674**	0.0882	0.106
fr2_360_kidnap	0.805	**0.799**	0.951	**0.971**	0.973	1.144
fr2_desk	1.33	1.65	**1.207**	1.64	1.81	**1.266**
fr2_desk_person	**0.412**	0.7	0.623	**0.434**	0.842	1.277
fr2_xyz	0.0653	**0.0619**	0.119	0.0808	**0.0772**	0.109
fr3_cabinet	**1.05**	1.08	-	1.57	**1.56**	-
fr3_large_cabinet	1.742	**1.731**	-	2.167	**2.1553**	2.93
fr3_long_office	**1.168**	1.180	1.233	**1.464**	1.512	1.577
fr3_nstr_ntex_far	**0.504**	0.677	-	**0.740**	0.876	-
fr3_nstr_tex_far	**0.575**	0.677	-	**0.662**	0.837	-
fr3_nstr_tex_loop	0.06	0.0597	**0.403**	**0.667**	0.08328	0.507
fr3_nstr_tex_sit_half	**0.160**	0.210	0.269	**0.178**	0.259	0.339
fr3_sit_xyz	**0.174**	0.187	0.343	**0.221**	0.24	0.413
fr3_str_notex_far	**0.865**	0.903	-	**0.946**	1.052	-
fr3_str_tex_far	**0.744**	0.790	0.892	**0.861**	0.917	0.988
fr3_walk_half	0.374	**0.354**	0.361	0.464	0.452	**0.448**
fr3_walk_xyz	0.275	**0.232**	-	0.345	**0.282**	-

TABLE I

RESULTS ON TUM RGBD DATASET. MEDIAN OVER 5 EXECUTIONS FOR EACH SEQUENCE IN TUM RGBD DATASET.

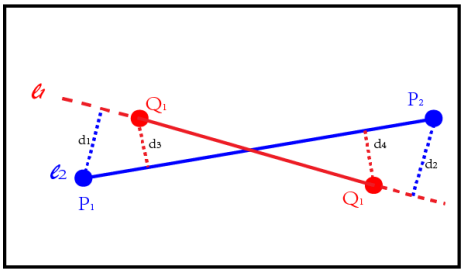


Fig. 9. 2D Line Segment Distance

divided into small patches. Finally, the points with the highest gradient value in each patch are selected as the keypoints. By this means, the selected points are uniformly distributed in the image.

2) IL Generation: As for line extraction, we use the public LSD segment detection algorithm [25]. The input image is first filtered using a Gaussian filter with a factor of 0.5. Then the lines are detected. The points too close to a line will be filtered out for the efficiency consideration. Here we adopt an adaptive distance threshold for each line. There may exist some lines overlapping with each other, which will cause redundant computation. Here we use an angle-grid based method to merge the similar lines. We select 5° as the grid size. The angles of all lines are normalized between 0° to 180° first and then put into the corresponding grids. In each grid, the lines with the smallest distance are merged.

The distance between two line segments is defined as follows. For each line segment, we first compute the distance from the endpoints of one line to the other line and sum the two point-to-line distance. After calculating this sum on each of the lines, we choose the smaller sum as the distance between these two line segment (Fig. 9):

$$D(l_1,l_2) = min(d_1+d_2, d_3+d_4). \tag{9}$$

After the IL extracted, the ILEPs and ILCPs are selected. The ILEP is selected as two endpoints, and the ILCPs are sampled uniformly on the IL. In some situations, a line exists in several keyframes. The fixed sampling interval usually results in the same sampling points in these keyframe. To utilizing more information, we use random sampling intervals.

TABLE II

RUNTIME RESULTS

(a) Tracking Runtime on TUM RGBD Dataset.

	Method	Tracking
TUM RGBD [26]	DMO-PL [5]	51.95 ms
	PL-SLAM [4]	51.06
	DSO [10]	7.283 ms
	Ours	**5.88 ms**

(b) Tracking Runtime on EuRoc Dataset.

	Method	Tracking
EuRoc [27]	LSD-SLAM [11]	27.6 ms
	ORB-SLAM [8]	17.9 ms
	S-PTAM [14]	47.3 ms
	LIBVISO2 [28]	24.8 ms
	DSO [10]	13.267 ms
	Ours	**9.549 ms**

VI. EXPERIMENT

In this section, we test our algorithm on various public datasets. We compare our method with two state-of-the-art SLAM systems: a indirect point-based method, ORB-SLAM [8] and a direct point-based method DSO [10].

A. TUM Mono Dataset [22]

We first compare these approaches on the TUM Mono Dataset [22]. This dataset provides 50 photometrically calibrated sequences. In other words, this dataset provides photometric calibration and exposure times for each sequence. We run each sequence for five times to reduce the influence of non-deterministic behavior. We will use two evaluation criterion: *alignment error* e_{align} and *accumulated*

rotational drift e_r. These two criteria reveal the error about translation and rotation, respectively. Due to our method is a visual odometry method and does not provide loop-closure detection and re-localization function, we disable explicit loop-closure detection and re-localization for ORB-SLAM. To make evaluation more precise, we do not enforce real-time execution to remove the dependency on the host machines CPU speed. For ORB-SLAM, we play the video at 20% speed. The results are shown in Fig. 6. From the figure, we can see that our method outperforms the ORB-SLAM and DSO both on *alignment error* e_{align} and *accumulated rotational drift* e_r on this dataset. In some situations, the ORB-SLAM tracks lost (with Inf error) whereas our method does not. Hence on this dataset, our method outperforms the ORB-SLAM and DSO on both accuracy and robustness.

B. TUM RGBD Dataset [26]

In many situations, we cannot get the photometric calibration and exposure times. So here we select the TUM RGBD Dataset to test the performance without photometric calibration and exposure times. We adopt the same setting as above. The DSO has the same setting as our method. As for our method and DSO, we omit photometric image correction and set related value zero. We use the *Absolute Trajectory Error (ATE)* and *Relative Pose Error (RPE)* as the evaluation. On this dataset, we choose the median over 5 executions on each sequence for a better comparison. The results are shown in TABLE I. From the table, we can see that ORB-SLAM gets the worst performance and fails in many sequences. The DSO gets a better performance than ORB-SLAM. As before, our method outperforms the other two methods in almost all sequences. As for efficiency, among the top five fast sequences, we can get 21.61 ms per frame. To get a fair comparison, we compute the mean time on all sequences. The result of tracking efficiency is shown in TABLE. II (a) withe some other approaches.

C. EuRoc Dataset

Besides above two datasets, the EuRoc dataset [27] is used to test efficiency further. The results are shown in TABLE II (b). Among the top five fast sequences, we can get 58.88 ms per frame. We can further improve the speed by increasing the point sampling interval on each line to reduce the extracted points.

REFERENCES

[1] D. Nistér, O. Naroditsky, and J. Bergen, "Visual odometry," in *Computer Vision and Pattern Recognition, 2004. CVPR 2004. Proceedings of the 2004 IEEE Computer Society Conference on*, vol. 1. Ieee, 2004, pp. I–I.

[2] S. Thrun and J. J. Leonard, "Simultaneous localization and mapping," in *Springer handbook of robotics*. Springer, 2008, pp. 871–889.

[3] J. Engel, J. Sturm, and D. Cremers, "Semi-dense visual odometry for a monocular camera," in *Proceedings of the IEEE international conference on computer vision*, 2013, pp. 1449–1456.

[4] 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. International Conference on Robotics and Automation (ICRA), IEEE*, 2017.

[5] S. Yang and S. Scherer, "Direct monocular odometry using points and lines," *arXiv preprint arXiv:1703.06380*, 2017.

[6] C. Mei, G. Sibley, M. Cummins, P. Newman, and I. Reid, "Rslam: A system for large-scale mapping in constant-time using stereo," *International journal of computer vision*, vol. 94, no. 2, pp. 198–214, 2011.

[7] Y. Lu and D. Song, "Robust rgb-d odometry using point and line features," in *Proceedings of the IEEE International Conference on Computer Vision*, 2015, pp. 3934–3942.

[8] R. Mur-Artal, J. M. M. Montiel, and J. D. Tardos, "Orb-slam: a versatile and accurate monocular slam system," *IEEE Transactions on Robotics*, vol. 31, no. 5, pp. 1147–1163, 2015.

[9] G. Klein and D. Murray, "Parallel tracking and mapping for small ar workspaces," in *Mixed and Augmented Reality, 2007. ISMAR 2007. 6th IEEE and ACM International Symposium on*. IEEE, 2007, pp. 225–234.

[10] J. Engel, V. Koltun, and D. Cremers, "Direct sparse odometry," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 2017.

[11] J. Engel, T. Schöps, and D. Cremers, "Lsd-slam: Large-scale direct monocular slam," in *European Conference on Computer Vision*. Springer, 2014, pp. 834–849.

[12] R. Gomez-Ojeda, F.-A. Moreno, D. Scaramuzza, and J. Gonzalez-Jimenez, "Pl-slam: a stereo slam system through the combination of points and line segments," *arXiv preprint arXiv:1705.09479*, 2017.

[13] R. Gomez-Ojeda and J. Gonzalez-Jimenez, "Robust stereo visual odometry through a probabilistic combination of points and line segments," in *Robotics and Automation (ICRA), 2016 IEEE International Conference on*. IEEE, 2016, pp. 2521–2526.

[14] T. Pire, T. Fischer, J. Civera, P. De Cristóforis, and J. J. Berlles, "Stereo parallel tracking and mapping for robot localization," in *Intelligent Robots and Systems (IROS), 2015 IEEE/RSJ International Conference on*. IEEE, 2015, pp. 1373–1378.

[15] A. J. Davison, I. D. Reid, N. D. Molton, and O. Stasse, "Monoslam: Real-time single camera slam," *IEEE transactions on pattern analysis and machine intelligence*, vol. 29, no. 6, pp. 1052–1067, 2007.

[16] E. Rublee, V. Rabaud, K. Konolige, and G. Bradski, "Orb: An efficient alternative to sift or surf," in *Computer Vision (ICCV), 2011 IEEE international conference on*. IEEE, 2011, pp. 2564–2571.

[17] H. Bay, A. Ess, T. Tuytelaars, and L. Van Gool, "Speeded-up robust features (surf)," *Computer vision and image understanding*, vol. 110, no. 3, pp. 346–359, 2008.

[18] V. Lepetit, F. Moreno-Noguer, and P. Fua, "Epnp: An accurate o (n) solution to the pnp problem," *International journal of computer vision*, vol. 81, no. 2, pp. 155–166, 2009.

[19] C. Forster, M. Pizzoli, and D. Scaramuzza, "Svo: Fast semi-direct monocular visual odometry," in *Robotics and Automation (ICRA), 2014 IEEE International Conference on*. IEEE, 2014, pp. 15–22.

[20] R. A. Newcombe, S. J. Lovegrove, and A. J. Davison, "Dtam: Dense tracking and mapping in real-time," in *Computer Vision (ICCV), 2011 IEEE International Conference on*. IEEE, 2011, pp. 2320–2327.

[21] J. Engel, J. Stückler, and D. Cremers, "Large-scale direct slam with stereo cameras," in *Intelligent Robots and Systems (IROS), 2015 IEEE/RSJ International Conference on*. IEEE, 2015, pp. 1935–1942.

[22] J. Engel, V. Usenko, and D. Cremers, "A photometrically calibrated benchmark for monocular visual odometry," *arXiv preprint arXiv:1607.02555*, 2016.

[23] J. Civera, A. J. Davison, and J. M. Montiel, "Inverse depth parametrization for monocular slam," *IEEE transactions on robotics*, vol. 24, no. 5, pp. 932–945, 2008.

[24] S. Leutenegger, S. Lynen, M. Bosse, R. Siegwart, and P. Furgale, "Keyframe-based visual–inertial odometry using nonlinear optimization," *The International Journal of Robotics Research*, vol. 34, no. 3, pp. 314–334, 2015.

[25] R. G. Von Gioi, J. Jakubowicz, J.-M. Morel, and G. Randall, "Lsd: A fast line segment detector with a false detection control," *IEEE transactions on pattern analysis and machine intelligence*, vol. 32, no. 4, pp. 722–732, 2010.

[26] J. Sturm, N. Engelhard, F. Endres, W. Burgard, and D. Cremers, "A benchmark for the evaluation of rgb-d slam systems," in *Intelligent Robots and Systems (IROS), 2012 IEEE/RSJ International Conference on*. IEEE, 2012, pp. 573–580.

[27] M. Burri, J. Nikolic, P. Gohl, T. Schneider, J. Rehder, S. Omari, M. W. Achtelik, and R. Siegwart, "The euroc micro aerial vehicle datasets," *The International Journal of Robotics Research*, vol. 35, no. 10, pp. 1157–1163, 2016.

[28] A. Geiger, J. Ziegler, and C. Stiller, "Stereoscan: Dense 3d reconstruction in real-time," in *Intelligent Vehicles Symposium (IV)*, 2011.