Direct Line Guidance Odometry

Shi-Jie Li¹, Bo Ren¹, Yun Liu¹, Ming-Ming Cheng¹, Duncan Frost², Victor Adrian Prisacariu²

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 lumiance) 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.

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
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.
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 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 Point) and line is denoted as IL (Immature Line). 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 Point) and line is denoted as IL (Immature Line).

C. Photometric Error Model

The photometric error is defined in a similar way as [10]. The photometric error of a point \( p \in \Omega \) in reference frame \( I_i \) observed in a target frame \( I_j \) is defined as:

\[
E_{p_j} := \sum_{p \in \mathcal{N}_p} w_p ||(t_j[p'] - b_j) - t_j e_{ai} t_i e_{bi}(I_i[p] - b_i)||_F,
\]

where \( \mathcal{N}_p \) is the set of pixels in the pattern, \( t_i \) and \( t_j \) are exposure times of the images \( I_i, I_j \) and \( ||*||_F \) is the Huber norm. The \( w_p \) is a gradient-dependent weight:

\[
w_p := \frac{c^2}{c^2 + ||I_i(p)||^2_L}.
\]

\( c \) is a fixed scalar and \( p' \) represents the projected point of \( p \) with inverse depth \( d_p \):

\[
p' = \pi_C(R \pi_C^{-1}(p, d_p) + t).
\]

where \( R \) and \( t \) represent the rotation and translation between the images \( I_i \) and \( I_j \), respectively. \( C \) denotes the intrinsic camera parameters. \( \pi_C : \mathbb{R}^3 \rightarrow \Omega \) is the projection from camera coordinate to the image plane using intrinsic parameters \( C \) and \( \pi_C^{-1} : \Omega \times \mathbb{R} \rightarrow \mathbb{R}^3 \) 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 \( l \in \mathcal{L} \) is the sum of point errors on it:

\[
E_{l_j} := \sum_{e \in \mathcal{E}} E_{e_j} + \sum_{p \in \mathcal{P}} E_{p_j},
\]

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

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.
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:

\[
p(p_p, p_l) = p(p_p|l) \cdot p(l)
\]

where \(p_p\) denotes the point position on a IL and \(p_l\) 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:

\[
E_{\text{photo}} := E_p + E_l \tag{6}
\]

where the \(E_p\) represents the total error on APs and \(E_l\) denotes the total error on ALs.

\[
E_p := \sum_{i \in \mathcal{F}} \sum_{p \in \mathcal{P}_i} \sum_{j \in \text{obs}(p)} E_{pj} \tag{7}
\]

\[
E_l := \sum_{i \in \mathcal{F}} \sum_{l \in \mathcal{L}_i} \sum_{j \in \text{obs}(l)} E_{lj} \tag{8}
\]

where \(i\) runs over all frames \(\mathcal{F}\), \(p\) over all APs \(\mathcal{P}_i\) in frame \(i\), and \(j\) over all frames \(\text{obs}(p)\) in which the AP \(p\) is visible. \(l\) over all ALs \(\mathcal{L}_i\) in frame \(i\), and \(j\) over all frames \(\text{obs}(l)\) in which the AL \(l\) is visible. The point error \(E_{pj}\) and the line error \(E_{lj}\) 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].
 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. 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.

**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.

**Fig. 8.** This figure shows the final map on sequence Machine Hall 02 in EuRoc MAV Dataset.
TABLE I
RESULTS ON TUM RGBD DATASET. MEDIAN OVER 5 EXECUTIONS FOR EACH SEQUENCE IN TUM RGBD DATASET.

<table>
<thead>
<tr>
<th>sequence</th>
<th>Absolute trajectory error (m)</th>
<th>Relative pose error (m)</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Ours</td>
<td>DSO</td>
</tr>
<tr>
<td>fr1_xyz</td>
<td>0.0538</td>
<td>0.0609</td>
</tr>
<tr>
<td>fr2_xyz</td>
<td>0.0653</td>
<td>0.0619</td>
</tr>
<tr>
<td>fr2_desk</td>
<td>0.0805</td>
<td>0.799</td>
</tr>
<tr>
<td>fr2_desk_person</td>
<td>0.412</td>
<td>0.7</td>
</tr>
<tr>
<td>fr2_office</td>
<td>0.419</td>
<td>0.959</td>
</tr>
<tr>
<td>fr3_cabinet</td>
<td>1.05</td>
<td>1.68</td>
</tr>
<tr>
<td>fr3_large_cabinet</td>
<td>1.792</td>
<td>1.731</td>
</tr>
<tr>
<td>fr3_long_office</td>
<td>0.504</td>
<td>0.677</td>
</tr>
<tr>
<td>fr3_nstr_tex_far</td>
<td>0.575</td>
<td>0.677</td>
</tr>
<tr>
<td>fr3_nstr_loop</td>
<td>0.06</td>
<td>0.0597</td>
</tr>
<tr>
<td>fr3_nstr_sit_half</td>
<td>0.160</td>
<td>0.210</td>
</tr>
<tr>
<td>fr3_sit_xyz</td>
<td>0.074</td>
<td>0.174</td>
</tr>
<tr>
<td>fr3_str_tex_far</td>
<td>0.865</td>
<td>0.903</td>
</tr>
<tr>
<td>fr3_str_sit_half</td>
<td>0.744</td>
<td>0.790</td>
</tr>
<tr>
<td>fr3_walk_half</td>
<td>0.374</td>
<td>0.354</td>
</tr>
<tr>
<td>fr3_walk_xyz</td>
<td>0.275</td>
<td>0.232</td>
</tr>
</tbody>
</table>

Fig. 9. 2D Line Segment Distance

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):

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

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.

<table>
<thead>
<tr>
<th>Method</th>
<th>Tracking</th>
</tr>
</thead>
<tbody>
<tr>
<td>TUM RGBD [26]</td>
<td></td>
</tr>
<tr>
<td>DMO-PL [5]</td>
<td>51.95 ms</td>
</tr>
<tr>
<td>PL-SLAM [4]</td>
<td>51.06 ms</td>
</tr>
<tr>
<td>DSO [10]</td>
<td>7.283 ms</td>
</tr>
<tr>
<td>Ours</td>
<td>5.88 ms</td>
</tr>
</tbody>
</table>

(b) Tracking Runtime on EuRoc Dataset.

<table>
<thead>
<tr>
<th>Method</th>
<th>Tracking</th>
</tr>
</thead>
<tbody>
<tr>
<td>EuRoc [27]</td>
<td></td>
</tr>
<tr>
<td>LSD-SLAM [11]</td>
<td>27.6 ms</td>
</tr>
<tr>
<td>ORB-SLAM [8]</td>
<td>17.9 ms</td>
</tr>
<tr>
<td>S-PTAM [14]</td>
<td>47.3 ms</td>
</tr>
<tr>
<td>LIBVIS2 [28]</td>
<td>24.8 ms</td>
</tr>
<tr>
<td>DSO [10]</td>
<td>13.26 ms</td>
</tr>
<tr>
<td>Ours</td>
<td>9.549 ms</td>
</tr>
</tbody>
</table>

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: an 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 $e_{\text{align}}$ 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) with 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