Direct Visual Odometry Using Lines for a Monocular Camera

Yingge Wang*, Zhiqiang Wang and Qing Zhu
Faculty of Information Technology Beijing University of Technology, Beijing, China
*Email: 18811713081@163.com

Abstract. Most of the visual odometry is based on the matching of the feature points, or the pixel matching of the direct method. However, images have another obvious feature, i.e. the line feature. If we use the point based visual odometry in low texture images, it may result in bad performance in the experiment because of few numbers of feature points. Although in some texture-less environments, it is still possible to reliably estimate the line based on geometric elements. For example, the structured edges are obvious in the indoor scenes. In this paper, we propose a monocular visual odometry method which is based on the combination of direct method and line feature. Then we use TUM-RGB and ICL-NUIM datasets to test our algorithm. Experimental results show that our method improves the robustness and accuracy of the estimation of the position and attitude of the camera.

1. Introduction
Simultaneous Localization and Mapping (SLAM) will be a key component of any autonomous robot system. Successful implementations of SLAM have generally been achieved with laser, sonar or stereo vision range sensors [1]. In this paper, we mainly talk about visual SLAM which solve location and map problems by using cameras, the sensor we used is monocular camera.

There are typically two categories of SLAM and VO approaches: (1) feature-based methods, such as ORB [2]. (2) Direct method [3],[4] have become popular in recently. The two methods both have their particular virtues.Line is a kind of important features in the images. In the artificial structured environment, line features are rich.The environment map constructed with line segment features has more intuitive geometric information. Line features has been used for RGB-D [5]. We choose some images with less-texture as tested pictures. We show the picture processed by the direct method of points and edges respectively in Figure. 1 and Figure. 2 We find out that it is may not work robustly by using points because of few point features in Figure. 1. But in Figure. 2, it can still detect many edges shown in red which could be used for estimating the position and pose of the monocular camera and mapping. We will prove this by various experiments in the following sections.
2. Proposed Method

2.1. Contributions
In this paper, we propose a monocular visual odometry method which is based on the combination of direct method and line feature. Line feature is applied to monocular direct method, especially for some texture-less environments. And the fusion error of the pair of points and lines is analyzed in this system. Our method outperforms or is comparable to existing direct VO in many datasets.

2.2. Method Outline
Our method estimates the current frame’s semi-dense inverse depth map. When we estimate the map, we use the high gradient pixels of the key frame. Our experiments are mainly divided into two threads. In the third and fourth chapters, we introduced tracking and mapping thread respectively. In the tracking thread, we firstly introduce the selection of key frames. Then we introduce the geometric and photometric error of the line features. In the fourth chapter, we introduce the line feature extraction and matching in the mapping thread and analyze the error. In the fifth chapter, we describe the implementation of the experiment and analyze the data obtained after the experiment.

3. Tracking

3.1. Overview
In the tracking thread, the identified depth map $D_r$ of the reference frame $I_r$ is assumed to be fixed. When a new image is input, there is a $SE(3)$ transformation relationship with the key frame by default. We need to get the current position and orientation information, and optimize this relationship based on the key frame and the current image data. It can be expressed as the following nonlinear least squares problems which is given by:

$$E(\hat{\xi}) = \sum_{i \in \Omega} r_i(\hat{\xi})^T \Sigma_i^{-1} r_i(\hat{\xi}) + \sum_{j \in M} g_j(\hat{\xi})^T \Sigma_g^{-1} g_j(\hat{\xi})$$  \hspace{1cm} (1)

where $\xi \in se(3)$ is an efficient optimization; $T \in SE(3)$ denotes the rigid transformation; $r(\xi)$ is the photometric residual; $g(\xi)$ is the line re-projection geometric error which emerged from line feature’s position observation; $r(i)$ represents photometric error has been defined by [4] as follow:

$$r_i = I_r(x_i) - I(\tau(x_i, D_r(x_i), \hat{\xi}))$$  \hspace{1cm} (2)

g(j) is the re-project error of pixel $x_j$ which is on the line $l_j$. $g(j)$ is formulated as follow:

$$g_j = I^\Lambda_j \tau(x_i, D_r(x_i), \hat{\xi})$$  \hspace{1cm} (3)

where $\tau(\cdot)$ is the homogeneous coordinate operation. This formula is used to calculate the pixels of edges in $I_r$ which also have a matching edge in current frame $I$. we use $\Sigma_r$ and $\Sigma_g$ representing the uncertainty of the photometric and re-projection error respectively.
The function (1) is an inconvenient direct solution of the least squares problem. Here we use Gauss-Newton method to optimize the solution of the incremental equation. After Gauss-Newton’s iteration \(n\), the following results are obtained:

\[
\delta\xi^n = -(J^TWJ)^{-1}J^TWE(\xi^n)
\]

where \(E = (r_1, ..., r_n, g_1, ..., g_m)^T\). \(J\) is the Jacobian of \(E\) \(wrt\ \xi\), \(W\) is the weight matrix computed from uncertainty \(\Sigma^{-1}\).

### 3.2. Reference Frame Selection

In our experiments, we use the method of key frame’s selection in [3]. It is an adaptive method, which is determined by the distance of motion. If the current camera motion is too far away from the reference frame, the current frame is created as a key frame. The distance function is given as follows:

\[
dist(\xi_{ji}) := \xi_{ji}^TW\xi_{ji}
\]

where \(\xi_{ji}\) is the \(SE(3)\) transformation representing a lie algebra form between the reference frame to the current frame; \(W\) is a weighted matrix; The distance threshold is determined by the average inverse depth of the current frame scene.

### 3.3. Tracking Uncertainty Analysis

In this part, we only give the analysis for re-project error uncertainty \(\Sigma_g\). The photometric error uncertainty \(\Sigma_r\) has been analysed in [4]. In the most case, a function \(f(x)\) output uncertainty propagated from the input uncertainty is expressed by:

\[
\Sigma_f \approx J_f \Sigma_x J_f^T
\]

In our system, line features are recorded as \(l_j = \chi_j \times x_j\). Firstly, we assume that the uncertainties \(\Sigma_p\) of the two line endpoint positions obey bi-dimensional Gaussians with \(\sigma = 1\). Then we compute the uncertainties of line equation coefficient \(l_j\) by using the Equation(6). We can compute the variance of reprojection point similarly. The final combination re-projection error covariance is obtained:

\[
\Sigma_{g_{ij}} = I_j^T \Sigma_{l_{ij}} I_j + x_j^T \Sigma_{\hat{l}_j} \hat{x}_j
\]

### 4. Mapping

#### 4.1. Numbering

In the mapping thread, we add line regularization to improve the accuracy of map based on the method of [4]. In [4], a semi-dense inverse depth map is estimated the depth of pixels with obvious gradient and expressed in inverse depth. It assumes that the inverse depth obeys Gauss distribution. Once a frame is selected as a key frame, the depth map of the reference frame is constructed with reference frame. After that, the new key frame will be triangulated to update the depth map. Assuming that the pose of the camera is fixed, the depth optimization formula is defined as follow:

\[
E(D) = \sum_i r_i (d)^T \Sigma_{r_i} r_i (d) + \sum_j G_j (d)^T \Sigma_{G_j} G_j (d)
\]

Where \(r_i\) is the stereo matching photometric error. \(G_j\) is edge regularization cost which represents the distance of edge pixel’s 3D point to 3D line. In the first term, the pixel in the image is independent. We use the regularization term \(G_j\) to make the depth of pixels on one edge correlate with each other. In our experiment, we improve system performance by optimizing \(r_i\) and \(G_j\).

#### 4.2. Detection and Stereo Match with Lines

The line feature is detected by LSD[6]. LSD is a kind of locally extracted line segment algorithm that can get sub-pixel accuracy in linear time. The core idea of LSD is that merging pixels with similar
gradient direction. As shown in the Figure 3, it is the original image. Firstly, LSD calculates the image’s down sampling and Gauss filtering to remove noisy interference as the Figure 4 shown, then calculates the gradient magnitude and direction of the image, also calculates the level-line angle of each pixel. Level-line field is composed of gradient of all pixels.

**Figure 3.** The original image.  
**Figure 4.** The level-line field image.  
**Figure 5.** The line support regions.

A series of line support regions can be obtained through region growing algorithm as shown in Figure 5. By counting n and k corresponding the number of pixels in the smallest circumscribed rectangle and the number of aligned points to determine whether the line support region is a straight line. This experiment applies the matching method to match the two edges \( l_1 \) and \( l_2 \) mentioned in [6].

### 4.3. Depth Calculation

We directly do line triangulation to compute all pixel’s depth together. We assume that the camera transformation of current frame \( I_c \) to \( I_r \) is \( R \in \text{SO}(3) \), then the 3D line can be represented the intersection of two back-projected plane:

\[
L = \begin{bmatrix}
\pi_1^T \\
\pi_2^T
\end{bmatrix} = \begin{bmatrix}
l_1^T K & 0 \\
l_2^T K & l_2^T K t
\end{bmatrix}
\]  
(9)

Where \( l_1 \) and \( l_2 \) are the line equation in \( I_r \) and \( I \) respectively. \( K \) is the camera intrinsic parameter. For each pixel, we can compute the intersection of the back-projected ray \( L \) with to get its depth. Figure 6 shows that how to compute the 3D line.

**Figure 6.** 3D line \( L \) could be computed by the intersection of two back-projected plane \( \pi \), \( \pi \).

### 4.4. Uncertainty Estimation

In this paper, our method uses a geometrical cost function based on the orthogonal distance between re-projected 3D lines and their measured end-points [8], defined as:

\[
d^2(x, I) = \frac{(x^T l)^2}{l_1^2 + l_2^2}
\]  
(10)

And the nonlinear error \( e \) is defined as follow:

\[
e = \left[ \frac{x^T l_1}{l_1^2 + l_2^2}, \frac{x^T l_2}{l_1^2 + l_2^2} \right]^T \quad \text{with} \quad l = [l_1, l_2]^T
\]  
(11)

### 4.5. 3D Line Regularization

The 3D points should lie on the same plane \( G \) after back-projected from the same 2D edge. In this section, we first create another coordinate frame \( F \) whose \( x \), \( y \) axis lie on the plane \( G \). We define the
transformed point which is on the new coordinate frame as $p_{n}$. We first use RANSAC to select a set of inlier 2D points. the uncertainty of the pixel to line Euclidean distance:

$$d_{mah} = \min_{p_{0}\in l} (p_{n} - p_{o})^{T}\Sigma^{-1}_{po} (p_{n} - p_{o})$$  \hspace{1cm} (12)$$

where $p_{0} \in l$ indicates a point lying on line $l$ in frame $F$. More details could be found in [9]. After RANSAC, we can find the largest consensus set of points $p_{0i}, i=1,...,n$. This becomes a 2D weighted line fitting problem and we want to find the best line $L^*$ so that:

$$L^* = \min_{L} \sum_{i} \delta(p_{n})^{2} \Sigma^{-1}_{p_{n}} \delta(p_{n})$$  \hspace{1cm} (13)$$

We can then transform the optimal $L^*$ in coordinate frame $F$ back to the original camera optical frame and determine the pixel depth on the line.

5. Result

In this section, our experiment is test on the datasets of TUM RGBD [9] and ICL-NUIM [10]. We use the relative position error metric (RPE) by Strum et al [11].To evaluate the performance of our method, we compare it with the ORB-SLAM front-end system with BA, SDVO, and two line-based VO experiments in [12] and [13]. The results of [12] and [13] have been given in their papers.

We use the fr3/long and fr2/xyz sequence in the TUM RGBD data set to test our algorithm. Table 1 shows the quantitative comparison of different methods, and our method achieves competitive results.

| Sequence    | Ours | ORB  | SDVO | [12] | [13] |
|-------------|------|------|------|------|------|
| fr3/long    | 1.80 | 0.7  | 2.1  | 6.9  | 0.28 |
| fr2/xyz     | 0.65 | 0.8  | 0.6  | 2.1  | 0.8  |

The following pictures shows visual results of our experiment and with the data in the table, example images in TUM datasets with varying texture, we can get the following analysis:

**Figure 7.** The result of one picture in fr3/long dataset after using our experiment.

**Figure 8.** The result of one picture in fr3/cabinet dataset.

**Figure 9.** Example of the result in fr3/structure dataset.

Since the picture of Figure 7 is rich in feature points, ORB with BA achieves best results. As there are also many line feature elements in the dataset environment, our algorithm also performs well. However, the calculation of line feature detecting and matching is too large, and the error will be too large, so our experimental results are not as good as ORB. Our experimental results are almost the same as those of SDVO. But it is better than [12] and [13] based on line feature VO. We show the results of other data sets in Table 2. The paper in [12] [13] did not give experimental results. The form ‘F’ stands for fail.

| Sequence     | Ours  | ORB  | SDVO |
|--------------|-------|------|------|
| ICL/office1  | 1.83  | F    | 1.33 |
| fr3/cabinet-big | 7.32  | 33.57| 16.23|


els is rather than feature points. So SDVO can perform better in these feature

As shown in Figure. 8 and Figure. 9, the pictures are feature-less. We can see that our algorithm achieves the best average result than the rest of others in Table 2. Scenarios in this table are feature-less sequences, and ORB does not perform particularly well or even fail to track feature points, such as in fr3/cabinet where ORB fails, because there are not many point features detected. But direct VO method SDVO still works to a certain extent because the direct method utilizing high gradient and edge pixels is rather than feature points. So SDVO can perform better in these feature-less environments than ORB. But our algorithm is significantly better than SDVO, this is because the object surface in the scene is almost uniform, and there is no large intensity gradient, so the SDVO photometric error minimization cannot work well, and our algorithm can still use the edge to minimize the edge re-projection error.

6. Conclusion
In this paper, we propose a monocular visual odometry method which is based on the combination of direct method and line feature. Our experiment is to add line features to the SDVO to enhance the robustness and accuracy of the whole system. Experimental results show that our algorithm can run in a texture-less environment and perform well in the test dataset.

References
[1] Huang A S, Bachrach A and Henry P, “Visual Odometry and Mapping for Autonomous Flight Using an RGB-D Camera,”IEEE International Symposium on Circuits & Systems. IEEE, 2017.
[2] Mur-Artal R, Montiel J M M and Tardós J D, “ORB-SLAM: A Versatile and Accurate Monocular SLAM System,” IEEE Transactions on Robotics, 2017, 33(5):1147-1163.
[3] Engel J, Schöps T and Cremers D, “LSD-SLAM: Large-Scale Direct Monocular SLAM,” 2014.
[4] Engel J and Cremers D, “Semi-dense Visual Odometry for a Monocular Camera,” IEEE International Conference on Computer Vision, 2014.
[5] Woo D M, Park D C and Han S S, “2D Line Matching Using Geometric and Intensity Data,” International Conference on Artificial Intelligence & Computational Intelligence, 2010.
[6] Gioi R G V, Jakubowicz J and Morel J M, “LSD: A Fast Line Segment Detector with a False Detection Control,” IEEE Transactions on Pattern Analysis & Machine Intelligence, 2010, 22(4):722-732.
[7] Hartley R and Zisserman A, “Multiple View Geometry in Computer Vision,” Multiple view geometry in computer vision / 2nd ed. 2003.
[8] Bartoli A and Sturm P, “The 3D line motion matrix and alignment of line reconstruction-: Computer Vision and Pattern Recognition,” 2001.CVPR 2001.Proceedings of the 2001 IEEE Computer Society Conference on, 2001.
[9] Yan L, Song D, “Robust RGB-D Odometry Using Point and Line Features,” IEEE International Conference on Computer Vision, 2015.
[10] Handa A, Whelan T and Mcdonald J, “A benchmark for RGB-D visual odometry, 3D reconstruction and SLAM,” IEEE International Conference on Robotics & Automation, 2014.
[11] Sturm J, Engelhard N and Endres F, “A benchmark for the evaluation of RGB-D SLAM systems,” IEEE/RSJ International Conference on Intelligent Robots & Systems, 2012.
[12] Handa A, Whelan T and Mcdonald J, “A benchmark for RGB-D visual odometry, 3D reconstruction and SLAM,” IEEE International Conference on Robotics & Automation, 2014.
[13] Tarrio J J and Pedro S, “Realtime Edge-Based Visual Odometry for a Monocular Camera,” IEEE International Conference on Computer Vision, 2016.