An Implementation of Monocular Visual Odometry

Mingxin Dong, Department of Mechanical Engineering, Texas A&M University
College Station, Texas, dmx-123@tamu.edu

***Abstract*—In this report, we introduced the classification and mathematical background for monocular visual odometry (MVO) algorithms. We then built a feature-based MVO model and tested its performance with the KITTI odometry data set. The location data we rendered is consistent with the ground truth of KITTI dataset. The processing speed of our MVO is 8.3 fps.**

Keywords—Visual Odometry, FAST, Lucas-Kanade, RANSAC, Nister's 5 Point Algorithm.

I. Introduction

Monocular visual odometry is designed for estimating robot movement and local map from a set of consecutive images recorded from a camera. It is considered to be important front-end framework of visual Simultaneous Localization and Mapping (v-SLAM) algorithms [1-2]. There are mainly two types of algorithms: the feature-based method [7] and the direct method [8]. Feature-based method follows protocol of tracking, mapping and loop closure detecting. For feature detection, methods including FAST, ORB [4, 7] and the conception of BRIEF have been developed. However, these calculations are often time-consuming, and sometimes dealing with only hundreds of feature points may lead to losing information. To avoid becoming overly-dependent on feature points, the direct method, which is based on the optical flow and greyscale invariance assumption, and thus having the ability to restore the dense structure, have been proposed. Furthermore, semi-direct tracking method SVO [3] has also been proposed. Adopting feature consistency hypothesis yet using direct method to avoid feature mapping, SVO showed a precession speed of 55 fps on ARM Cortex A9 (1.6GHz CPU).

II. Background

We adopted a feature-based algorithm with optical flow method to boost simulation efficiency, for it has been noticed that optical flow method also has the characteristics of tracking feature points [9]. In this section, we will briefly illustrate the mathematics behind.

A. Monocular Camera Model

By defining pixel coordinate system, we could record and describe the real-world system digitally. Let the camera coordinate of P be [X, Y, Z], its image P' [X', Y', Z'], and let the pixel coordinate of P be [u, v]. Suppose that pixel coordinate zoomed in/out in u/v axis direction by a factor of α/β; and suppose the origin shifted $[c_x,\ c_y]$. Then we have:

$$u = \alpha X' + c_x = f_x \frac{X}{Z} + c_x \tag{1}$$

$$v = \beta Y' + c_y = f_y \frac{Y}{Z} + c_y \tag{2}$$

$$Z\begin{bmatrix} u \\ v \\ 1 \end{bmatrix} = \begin{bmatrix} f_x & 0 & c_x \\ 0 & f_y & c_y \\ 0 & 0 & 1 \end{bmatrix}\begin{bmatrix} X \\ Y \\ Z \end{bmatrix} = KP \tag{3}$$

where K is named the camera intrinsic matrix. Considering the motion of camera, we need to add the transformation of P from the world coordinate to camera coordinate. Let P_W be the world coordinate of camera, then $P = TP_w = RP_w + t$. R, t is defined to be the camera extrinsic.

In actual photographing, distortion can happen in radial and tangential directions due to the geometry of lens and camera design. Radial distortion can be represented as follows:

$$rd_{x,y} = coord_{x,y} \cdot (1 + k_1 r^2 + k_2 r^4 + k_3 r^6) \tag{4}$$

Tangential distortion could be represented as:

$$x_{td} = x + [2p_1 xy + p_2(r^2 + 2x^2)] \tag{5}$$

$$y_{td} = y + [p_1(r^2 + 2y^2) + 2p_2 xy] \tag{6}$$

The distortion coefficient is thus given by $[k_1,\ k_2,\ p_1,\ p_2,\ k_3]$.

B. Feature Detection

We use the FAST [4] corner detector for feature detection. Suppose there is a point P which pixel p and grayscale I_p. To test whether P is a corner point or not, we follow such protocol:

- Choose 16 pixels that fall on a circle of radius 3 and centered on pixel p, as shown in Fig. 1.
- Set a threshold value T. If there exist consecutive N points (usually N = 12) with grayscale $[I_p - T,\ I_p + T]$, then we say pixel p is a corner point.

To boost detecting efficiency, usually we detect pixel 1, 9, 5, 13 as pretest. If p is a corner, then at least three of these must all be brighter than $I_p + T$ or darker than $I_p - T$. In addition, when we need to detect multiple feature points in adjacent locations, non-maximum suppression shall be applied.

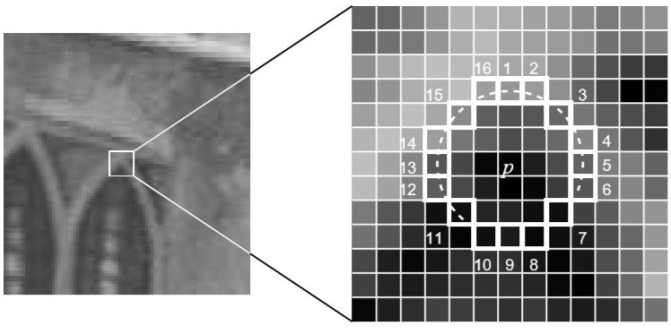

Fig 1. Image illustrating FAST algorithm taken from original paper [4].

C. Feature Tracking

We use Lucas-Kanade [5] feature tracker for finding sparse pixel-wise correspondences. In LK algorithm, the grayscale of an image can be described by pixel coordinates and time, namely $I(x, y, t)$. The problem then become mathematical:

$$min_{\Delta x,\Delta y} = \|I_1(x, y) - I_2(x + \Delta x, y + \Delta y)\|_2^2 \quad (7)$$

Assuming that the pixel greyscale value for a specific point remain the same in a set of frames, and that adjacent pixels within a window have similar motion, we obtain:

$$\frac{\partial I}{\partial x}dx + \frac{\partial I}{\partial y}dy + \frac{\partial I}{\partial t}dt = 0 \quad (8)$$

$$\frac{\partial I}{\partial x}\frac{dx}{dt} + \frac{\partial I}{\partial y}\frac{dy}{dt} = -\frac{\partial I}{\partial t} \quad (9)$$

Consider a window with w^2 pixels, then we will have a set of equations descripting the relation of pixel speed, pixel grayscale gradient and grayscale change over time:

$$[I_x \quad I_y]_k \begin{bmatrix} u \\ v \end{bmatrix} = -I_{tk} \quad (10)$$

Solving this set of over-determined equations by ordinary least squares method gives us:

$$\begin{bmatrix} u \\ v \end{bmatrix}^* = -(A^T A)^{-1} A^T b \quad (11)$$

$$A = [\,[I_x \;\; I_y]_1 \quad \cdots \quad [I_x \;\; I_y]_k\,]^T \quad (12)$$

$$b = [I_{t1} \quad \cdots \quad I_{tk}]^T \quad (13)$$

We also applied image pyramid to enhance the robustness of our tracking algorithm.

D. Pose Estimation

Once we have point-correspondences, we apply epi-polar constraint technique to estimate camera motion. For a point in space P, its pixel coordinates in two frames p_1, p_2 with scale s_1, s_2, we have:

$$s_1 p_1 = KP \qquad s_2 p_2 = K(RP + t) \quad (14)$$

where K denotes the intrinsic matrix, and R, t the extrinsic of our camera. In homogeneous coordinate, $sp \cong p$. Now let $x_2 \cong Rx_1 + t$, $x_{1,2} = K^{-1}p_{1,2}$, we should have

$$x_2^T t^\wedge x_2 \cong x_2^T t^\wedge R x_1 = 0 \quad (15)$$

Plugging in p_1, p_2, we have:

$$p_2^T K^{-T} t^\wedge R K^{-1} p_1 = 0 \quad (16)$$

By defining essential matrix $E = t^\wedge R$ or fundamental matrix $F = K^{-T}EK^{-1}$, we could solve for R, t from E or F directly.

To solve for pose information R, t from essential matrix E, we adopt SVD method, then we have

$$E = U\Sigma V^T \quad (17)$$

where U, V are orthogonal matrices, and Σ is singular value matrix. We shall have 4 sets of poses (R, t):

$$R_{1,2} = U \cdot R_Z^T(\pm\tfrac{\pi}{2}) \cdot V^T \quad (18)$$

$$t_{1,2} = U \cdot R_Z(\pm\tfrac{\pi}{2}) \cdot \Sigma U^T \quad (19)$$

The final pose (R, t) is determined by the fact that the depth of point P in camera should always be positive.

Next, we adopt Nister's 5-point algorithm [6] to solve for the essential matrix. We represent essential matrix with a column vector e, and put all 5 pair of points in one matrix U such that:

$$U_{5\times 9} \cdot e_{9\times 1} = 0 \quad (20)$$

Let [x, y, z, w] be 4 column vectors that span the null space of $U_{5\times 9}$, then the solution of E should be the linear combination of their corresponding matrices:

$$E = c_x X + c_y Y + c_z Z + W \quad (21)$$

With constraints $\det(E) = 0$, and

$$EE^T E - \tfrac{1}{2} trace(EE^T)E = 0 \quad (22)$$

Then applying Gauss-Jordan elimination method would lead us to the final result.

Sometimes, there are feature points that mismatch with their correspondence, causing trouble in estimation of pose and triangulation. A standard technique of handling such situation is RANSAC, which is accomplished with the following steps:

- Select random sample of minimum required size to fit model parameters and compute a putative model from sample set.
- Compute the set of inliers to this model from whole dataset and check if current hypothesis is better than any other of the previously verified.
- Repeating steps 1-2 for a prescribed number of iterations.

III. IMPLEMENTATION

The MVO we designed mainly consisted of several modules as listed below:

- Read and undistort 2 adjacent (grayscale) images from camera, I_t and I_{t+1}. Parameters such as intrinsic matrix could be obtained via OpenCV photo calibration.
- Use FAST algorithm to detect features from current (reference) frame, and track those features to new frame. If the number of features drop below a certain threshold (n = 2000), start over with new detection.
- Use Nister's 5-point algorithm with RANSAC to compute the essential matrix. Then estimate pose information R, t from the essential matrix.
- Estimate the scaling factor from external source (i.e., ground truth information in datasets), and concatenate the translation vectors, and rotation matrices.

IV. RESULTS

We have chosen one of the most challenging sequences of the KITTI visual odometry benchmark (sequence 00) to testify our algorithm. From the comparison shown in Fig. 2, it could be observed that the local mapping information we calculated is consistent with the ground truth information provided in the

Fig. 2. (a). KITTI dataset ground truth of sequence 00. (b). The result of our MVO simulation (from right camera). (c). A comparison between (a) and (b).

KITTI dataset. The cumulative error could not be eliminated without backend optimization, but generally speaking, the performance of our MVO is satisfactory for a single loop.

The processing time for the whole sequence (4540 frames) is around 9 minutes, therefore it took about 0.12s on average for our MVO to process one single frame. The program is compiled serially, with an Intel i7-9750H (2.6GHz CPU).

V. SUMMARY AND FUTURE WORK

In summary, a monocular visual odometry algorithm and its running results on the KITTI dataset have been presented. The closed-loop result remained within reasonable error bounds. The image processing speed is approximately 8.3 fps.

For future work, we plan to add backend modules to our visual odometry. The backend should include optimization (bundle adjustment), loop closure detection, map construction and storage.

REFERENCES

[1] F. Fraundorfer, D. Scaramuzza. Visual Odometry: Part 1. IEEE Robotics and Automation Magazine, 2011.

[2] D. Scaramuzza, F. Fraundorfer. Visual Odometry: Part 2. IEEE Robotics and Automation Magazine, 2012.

[3] C. Forster, M. Pizzoli, and D. Scaramuzza, "SVO: Fast semi-direct monocular visual odometry," In IEEE International Conference on Robotics and Automation (ICRA), 2014.

[4] T. Drummond, E. Rosten, "Machine learning for high-speed corner detection," In European Conference on Computer Vision, 2006.

[5] B. D. Lucas, T. Kanade, "An Iterative Image Registration Technique with an Application to Stereo Vision," In Proceedings of the 7th International Joint Conference on Artificial Intelligence, 1981.

[6] D. Nister, "An Efficient Solution to the Five-Point Relative Pose Problem," IEEE Transactions on Pattern Analysis and Machine Intelligence, June 2004, 26(6): 756-770.

[7] R. Mur-Artal, J. D. Tardós, "ORB-SLAM2: An Open-source SLAM System for Monocular, Stereo, and RGB-D cameras," IEEE Transactions on Robotics, 2016, 33 (5): 1255-1262.

[8] J. Engle, T. Schöps, D. Cremer, "LSD-SLAM: Large-scale Direct Monocular SLAM," Proceedings of European Conference on Computer Vision, 2014: 834-849.

[9] B. Kitt, A. Geiger, H. Lategahn, "Visual Odometry Based on Stereo Image Sequences with RANSAC-based Outlier Rejection Scheme," Proceedings of Intelligent Vehicles Symposium, 2010: 486-492.

p.s. Github repository of this project:
https://github.com/dmx-123/robotics-project