Optical Flow based Monocular Visual Odometry Initialization

Jincheng Li
jcli40@stanford.edu

Abstract

Visual odometry is an active research field in robotics. In this paper, the method of using only optical flow tracked features to initialize monocular camera pose for visual odometry is discussed. The discussed algorithms are experimented on EuRoC MAV dataset and results are analyzed.

1. Introduction

Localization is a fundamental pre-requisite in robotic systems to enable autonomous navigation. There exist various sensors to capture ego motion to help recover past trajectory thus identify robot location, from passive measurement sensors such as wheel-based odometry, GPS, IMU and camera, to active ones like radar, lidar.

Passive sensors are in general preferable as active sensors might suffer from interference in crowded circumstances. Among all the passive sensors, wheel-based odometry can only provide 1-dimension result while high-accuracy GPS/IMU systems come with expensive price tag. In contrast, camera, which is becoming more and more popular in recent years in robotics systems such as autonomous driving cars, is a very affordable passive sensor yet provides rich information of surrounding environment.

Taking advantage of the rich information from one or more cameras to estimate ego position and/or, to identify robot trajectory is an active research field called VO (Visual Odometry). VO is an essential part of another active research field called V-SLAM (Visual Simultaneously Localization and Mapping).

1.1. Visual Odometry

The term VO won popularity since 2004 in Nister's landmark article [9]. Just like the vehicle wheel odometry incrementally accumulates wheel rotations to estimate ego motion, VO examines images captured by on-board camera(s) to recover relative poses of robot body when the images are taken. Stitching recovered poses incrementally provides the trajectory, as well as location of the moving robot.

The problem of VO is similar in many ways to SfM (Structure from Motion) and SLAM problems. However, unlike SfM problem which usually deals with huge computation-intensive tasks offline, VO or SLAM system requires real-time responses to direct robot motion. VO can be deemed as a subset of SLAM system, i.e., the localization part. It doesn't need to maintain global structure/map consistency as required and fulfilled by loop closure in SLAM. VO only cares about local consistency.

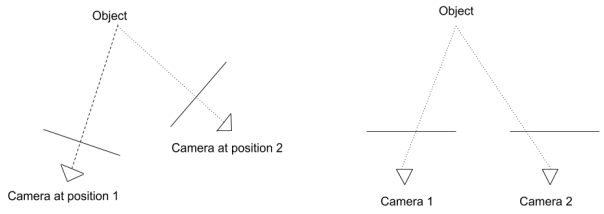

Figure 1. VO camera sensor setups. (Left) Monocular camera sensor captures images at different positions. (Right) Stereo camera sensor captures two images simultaneously at fixed distance t, the baseline.

1.2. Monocular and Stereo Camera System

Two popular camera sensor set-ups exist today in VO research: monocular and stereo camera systems, as shown in Fgiure 1.

Monocular camera system contains a single camera which captures images sequentially as robot body, where it attaches, moves. In comparison, stereo camera system setup rigidly holds two cameras which usually face same direction in parallel but at fixed distance t, called the baseline.

Although both setups follow same epipolar geometry described in [4] to recover relative pose, stereo cameras can restore absolute scale according to baseline distance while monocular system cannot recover from scale ambiguity. Nonetheless, due to limited baseline length in real-life applications, stereo camera setup degenerates to monocular case when object distance D to baseline t ratio

$$ratio = D/t \tag{1}$$

increases significantly. Thus this paper focuses on monocular discussion without losing generality.

1.3. EuRoC MAV dataset

The EuRoC MAV dataset [1] is used to evaluate the VO algorithm described in this paper. The dataset is chosen because:
1) It is newly created in 2015 together with ground truth.
2) The camera sensors are global-shuttered thus potential issues with rolling-shuttered cameras are completed avoided.
3) Time-synchronized data from other sensors allow future extension of the project.

Although EuRoC dataset contains stereo camera images, we will only use cam0 data as the monocular sensor captures in this paper.

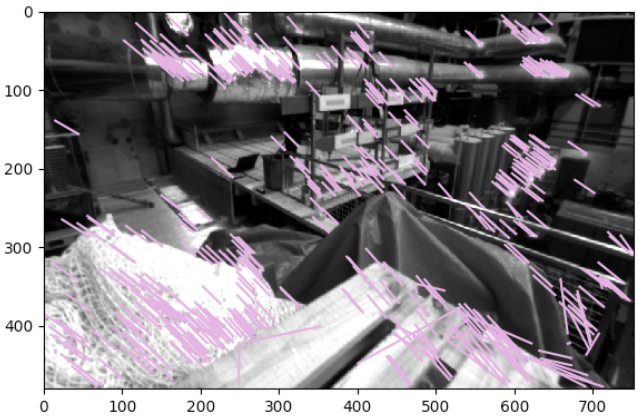

Figure 2. OpenCV optical flow features tracking example on EuRoC MAV dataset.

2. Algorithms Overview

The task of a VO system is to estimate relative poses between consecutive image frames temporally captured by the same camera sensor rigidly mounted on robot body.

Sequentially concatenating the estimated poses provides the current location and orientation of camera. Since relative pose is known from robot body to camera sensor, the robot location and orientation can be easily calculated from camera ones. Usually the camera center of first selected frame is chosen to be the world coordinate system origin.

The steps of optical-flow based monocular VO described in this paper are summarized in Table 1 in simplest format. Due to time constraint, key elements usually exist in VO systems, such as key-frame maintenance, tracking lost recovery etc. are not included in this paper. This paper focuses on monocular initialization only instead of complete VO, although all steps in Table 1 are covered.

The key algorithms involved are detailed in next section.

3. Detailed Technical Approach

The whole process is depicted in Figure 4.

A.	Camera images undistortion assuming calibrated camera.
B.	Use optical flow tracking to establish 2D-2D feature points correspondences between consecutive frames.
C.	Estimate Fundamental or Essential Matrix using epipolar geometry constraints. Use RANSAC [2] methods to rule out outliers in point-pairs.
D.	Recover camera pose (rotation R and translation T) from chosen Fundamental or Essential Matrix estimation, minimizing re-projection error to successfully initialize the VO.
E.	Triangulate 2D feature points to recover feature 3D coordinates in world coordinate system.
F.	Continue to use optical flow to track features and use triangulation to establish 3D-2D mapping problem to recover follow-up R and T poses using PnP algorithm.
G.	Camera trajectory can be obtained by stitching recovered camera poses, which is output of the VO program. Compare with ground truth and results analysis.

Table 1. Optical flow based Monocular VO steps.

3.1. Image Distortion Correction

Image frames captured by real-life cameras suffer from lens radial and tangential distortions. Images are undistorted using camera parameters acquired in calibration process, assuming below distortion model [10], k_1, k_2, p_1, p_2:

$$x_{radial} = x(1 + k_1 r^2 + k_2 r^4)$$
$$y_{radial} = y(1 + k_1 r^2 + k_2 r^4)$$
$$x_{tangential} = x + [2p_1 xy + p_2(r^2 + 2x^2)]$$
$$y_{tangential} = y + [p_1(r^2 + 2y^2) + 2p_2 xy]$$

3.2. Optical Flow based Feature Tracking

Optical flow based on Lucas-Kanade algorithm [6] is a popular method to find and track features in consecutive image frames. It is relatively cheap in computation comparing to descriptor based methods, such as SIFT (Scale-invariant Feature Transform) [5].

In this paper, OpenCV optical flow implementations [11] are used for experiments. Figure 2 shows an example of OpenCV optical flow tracking feature points on EuRoC MAV dataset. Tracking outliers are easily found there.

3.3. Monocular VO Initialization

Monocular VO initialization runs only once in a successful odometry recording process. It determines the origin and

three axis of world coordinate system (usually set at camera center of very first image frame as illustrated in Figure 3), as well as initial features (X, Y, Z) coordinates in this world system besides the scale in the world system for follow up operations.

3.3.1 Fundamental F or Essential Matrix E, 2D-2D correspondence

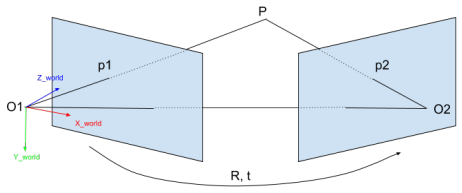

Figure 3. Epipolar constraint illustration.

Let P be a feature point in world system, p_1, p_2 be the correspondences of P's images captured on image plane of monocular camera sensor at different times. We have the epipolar constraint [4]:

$$p_2^T F p_1 = 0 \tag{2}$$

F is the Fundamental Matrix. Since camera is calibrated thus intrinsics K is known, equation 2 can be re-written using E, Essential Matrix:

$$p_2'^T E p_1' = 0 \tag{3}$$

where $E = K^T F K$, p_1' and p_2' are now coordinates on canonical image plane.

3.3.2 Extract R, t from E and Triangulation

Using the algorithm described in section 9.6 of [4], we can extract relative rotation and translation R, t, between two camera positions from E. We have to triangulate a sample point into 3D space to find the true solution from four solutions provided.

3.3.3 Initialization with RANSAC

Feature correspondences in 2D image, p_1, p_2, are provided by optical flow tracker described in 3.2. We need at least 8 point correspondences to calculate F (normalized 8-pt algorithm [4]), or at least 5 point correspondences to calculate E [8]. In this paper we use the linear 8-point algorithm for its simpler implementation.

We can calculate least squares solution of F as we have more than 8 point feature correspondences. However, due to existing outliers (easily seen in Figure 2), we'll get erroneous result. Here, we use RANSAC [2] algorithm to randomly select 8 point correspondences to calculate a candidate F_i. Then we calculate E_i according to candidate F_i. Pose candidate R_i, t_i follows.

Finally, using K, R_i, t_i, we re-project *ALL* feature correspondence points one by one onto image plane and examine the re-projection error, mean of L-2 distance. If error is larger than certain threshold $ThreshErr$ (set to 2 pixels), the feature correspondence pair is deemed as outlier for current candidate F_i, R_i, t_i.

$$err_{reproj} = \frac{\sum_1^n L2\_distance(x_i, \hat{x_i})}{n} \tag{4}$$

$$if(err_{reproj} < ThreshErr) : inlier_{cnt} += 1 \tag{5}$$

In the meantime, a min_err variable is maintained to keep record of minimum re-projection error seen so far in a successful iteration of RANSAC process. A successful iteration has ratio between number of inliers and total points above another threshold $ThreshRatio$ (set to 0.6). The final solution is determined by the successful iteration with minimum re-projection error.

All inlier points are then triangulated into world system to get (X,Y,Z) coordinates for future steps. They form the initialized point cloud of features. The camera coordinate system at the first camera when VO is successfully initialized is determined to be the world coordinate system, as shown in Figure 3.

3.4. 3D-2D point-pairs: PnP

Once initial point cloud is available, continue using optical flow helps set up 2D-3D correspondences that we can ever since use to solve R and t for later incoming frames.

A 2D-3D point pair satisfies the following equation:

$$s \begin{bmatrix} u_1 \\ v_1 \\ 1 \end{bmatrix} = \begin{bmatrix} t_1 & t_2 & t_3 & t_4 \\ t_5 & t_6 & t_7 & t_8 \\ t_9 & t_{10} & t_{11} & t_{12} \end{bmatrix} P \tag{6}$$

u_1, v_1 are coordinates on canonical image plane, $P = (X, Y, Z, 1)^T$ is the corresponding 3D point, s is the scale

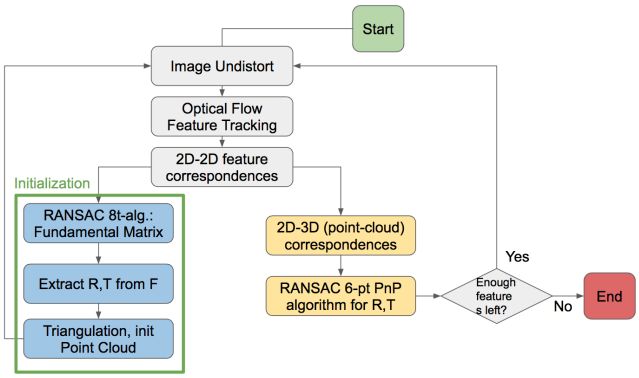

Figure 4. Monocular VO Initialization program diagram.

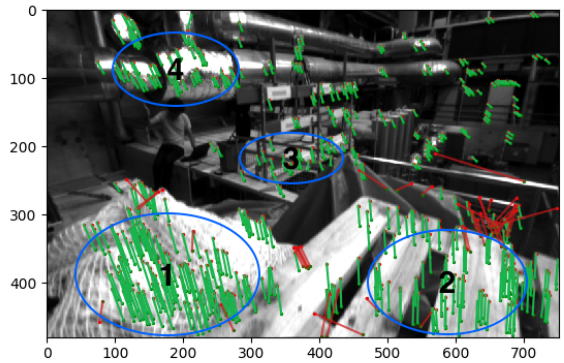

Figure 5. RANSAC-based 8 point algorithm rules out outliers (RED) in optical flows. Inliers (GREEN) are kept.

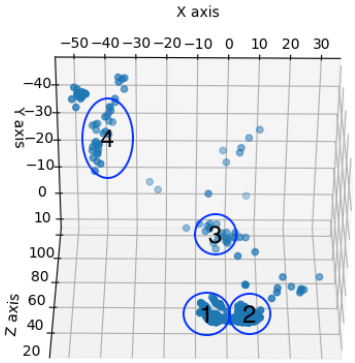

Figure 6. Initialized point cloud example. Groups match those circled out in Figure 5

ambiguity. According to 6-point linear PnP algorithm described in chapter 7.7.1 of [3] we can set up equation:

$$\begin{bmatrix} {P_i}^T & 0 & -u_i {P_i}^T \\ 0 & {P_i}^T & -v_i {P_i}^T \end{bmatrix} \begin{bmatrix} T_1 \\ T_2 \\ T_3 \end{bmatrix} = 0 \qquad (7)$$

where $T_1 = (t_1, t_2, t_3, t_4)^T, T_2 = (t_5, t_6, t_7, t_8)^T, T_3 = (t_9, t_{10}, t_{11}, t_{12})^T$ represent 12 unknowns. Randomly picking 6 pairs of 2D-3D correspondences, each provides two equations, provides a solution. Again, RANSAC and re-projection error are used here to finally determine the best solution.

From the solution of 3x4 T matrix, using SVD can recover rotation and translation respectively. The details are omitted here.

3.5. Exit

The process continues for each new incoming image frame that optical flow tracker sets up 2D-3D relationship of correspondences which is followed by rotation and translation matrices recovery. As optical flow will not successfully track all features and our VO initialization algorithm keeps removing outliers, we lose features after every iteration. The process ends when either of two conditions is met:

1) Successfully tracked features die out under certain threshold (set to 200 points), or

2) Accumulated re-projection error goes beyond a certain threshold (set to 0.8 pixel).

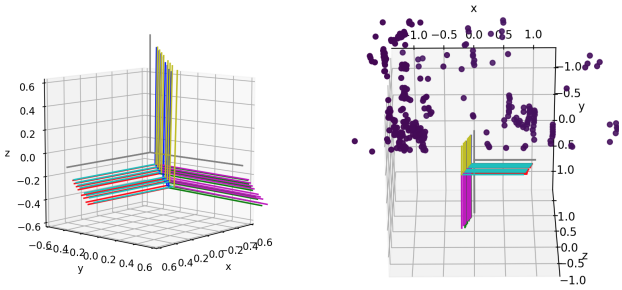

Figure 7. Estimate vs. ground-truth in world coordinate system, and point cloud. Gray axises is the world system which is the camera system of very first frame when the VO initialization successfully started.

4. Result Analysis

4.1. RANSAC

RANSAC algorithm proves it can successfully rule out outliers (marked RED) and keep only inliers (marked GREEN) as shown in Figure 5.

Initialized point cloud calculated from above described algorithm is illustrated in Figure 6. Please be noticed that monocular VO cannot recover absolute scale, so the axis scales are only showing relative locations of each point in the cloud up to a scale.

4.2. Pose/Trajectory Recovery and Error Analysis

The algorithm proves it can successful recover ego poses and trajectory before exit. As illustrated in Figure 7, estimation (Red - X axis, Green - Y axis, Blue - Z axis) is very close to ground truth (Cyan - X axis, Magenta - Y axis, Yellow - Z axis).

The data associated with this successful VO initialization example is logged in Figure 8. As expected, the survived

2D-2D init facts:

frame	PCL mean depth	ave depth	Mean reproj err	Inliers count	Total features
3004	44.91017625	55.56503662	0.1419504	584	594

2D-3D facts:

frame	Mean reproj err	Inliers count	Total features
3005	0.161571564	582	583
3006	0.196699347	582	582
3007	0.297819766	582	582
3008	0.375696309	579	582
3009	0.426948351	576	579
3010	0.473943616	576	576
3011	0.436955965	572	576
3012	0.556824174	567	572
3013	0.607335422	553	567
3014	0.621218022	547	549
3015	0.712867157	544	547
3016	0.689781693	535	544
3017	0.911043449	378	535

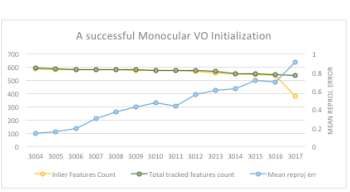

Figure 8. Data associated with an example successful VO initialization process.

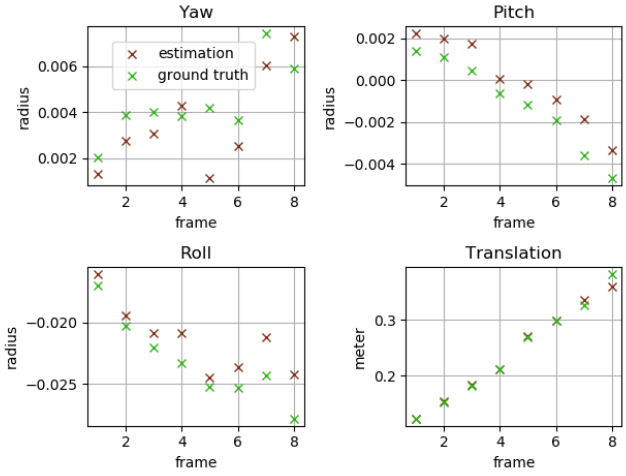

Figure 9. Estimate vs. ground-truth in Euler-Angles and Translation.

features going through optical flow tracking, then RANSAC process, become less and less after every iteration, in contrast, the accumulated re-projection error keeps increasing until be above threshold at program termination.

Errors of estimation vs. ground truth in terms of Yaw/Pitch/Roll Euler angles for rotation and distance errors for translation in this example are listed in Figure 9, which explains the VO initialization success we see in Figure 7.

4.3. Failure Cases

There exist cases when the proposed algorithm doesn't work.

One such failure case is illustrated in Figure 10 where all tracked features happen to lie on same plane, which suffers from degeneration of Fundamental matrix. Homography matrix is the solution for this case as used in ORB-SLAM [7].

Other failure cases include static-scene (noise-dominant calculation), or pure rotation (another F matrix degeneration case) etc.

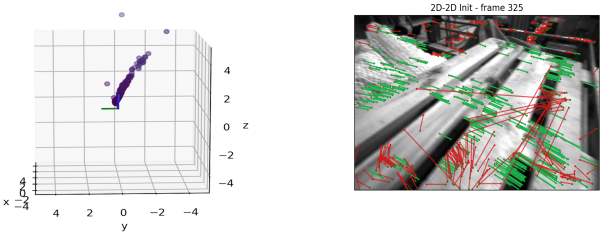

Figure 10. Fundamental matrix degenerates when all feature points lie on same plane. Need resort to Homography Matrix in such case.

5. Conclusion and Future Work

In this paper, we demonstrate an algorithm to initialize a monocular visual odometry system purely based on optical flow features tracking instead of feature matching methods such as SIFT/SURF, and geometry methods to recover rotation and translation matrices from 2D-2D/2D-3D correspondences. The algorithm is successfully applied on EuRoC dataset. The error accumulation and failure cases are analyzed. The source code is available at https://github.com/jinchenglee/Monocular_VO_on_EuRoC_dataset.

Future work includes:

1. Modify code to be OOP coding style. Due to strict time schedule of the course report, the source code is implemented as dataflow-driven.
2. Add Homography matrix based initialization to fix failure cases.
3. Use Bundle Adjustment methods to maintain key-frame and local map to reduce error accumulation, which paves the way for a complete VO implementation.

References

[1] 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*, 2016.

[2] M. A. Fischler and R. C. Bolles. Random sample consensus: A paradigm for model fitting with applications to image analysis and automated cartography. *Commun. ACM*, 24(6):381–395, June 1981.

[3] X. Gao, T. Zhang, Y. Liu, and Q. Yan. *14 Lectures on Visual SLAM: From Theory to Practice*. Publishing House of Electronics Industry, 2017.

[4] R. I. Hartley and A. Zisserman. *Multiple View Geometry in Computer Vision*. Cambridge University Press, ISBN: 0521540518, second edition, 2004.

[5] D. G. Lowe. Distinctive image features from scale-invariant keypoints. *International Journal of Computer Vision*, 60:91–110, 2004.

[6] B. D. Lucas and T. Kanade. An iterative image registration technique with an application to stereo vision (ijcai). In *Pro-*

ceedings of the 7th International Joint Conference on Artificial Intelligence (IJCAI '81), pages 674–679, April 1981.

[7] R. Mur-Artal, J. M. M. Montiel, and J. D. Tardós. ORB-SLAM: a versatile and accurate monocular SLAM system. *CoRR*, abs/1502.00956, 2015.

[8] D. Nistér. An efficient solution to the five-point relative pose problem. *IEEE Trans. Pattern Anal. Mach. Intell.*, 26(6):756–777, June 2004.

[9] D. Nistér, O. Naroditsky, and J. R. Bergen. Visual odometry. *Proceedings of the 2004 IEEE Computer Society Conference on Computer Vision and Pattern Recognition, 2004. CVPR 2004.*, 1:I–I, 2004.

[10] OpenCV. Opencv camera calibration undistortion document. https://docs.opencv.org/3.1.0/d4/d94/tutorial_camera_calibration.html.

[11] OpenCV. Opencv optical flow document. https://docs.opencv.org/3.1.0/d7/d8b/tutorial_py_lucas_kanade.html.