Contents

Abstract

This report explains a monocular visual odometry algorithm, which has been implemented in OpenCV/C++. We report results on the KITTI dataset (using only one image of the stereo dataset). Since it is a monocular implementation, we cannot do absolute scale estimation, and thus that quantity is used from the ground truths that we have.

1 Introduction

Visual Odometry is the estimation of 6-DOF trajectory followed by a moving agent, based on input from a camera rigidly attached to the body of the agent. The camera might be monocular, or a couple of cameras might be used to form a stereo rig. In the monocular approach, it is not possible to obtain the absolute scale of the trajectory, while it is certainly possible to do so for the stereo approach, assuming we know the baseline (distance between the two cameras of the stereo rig).
Visual Odometry has attracted a lot of research in the recent years, with new state-of-the-art approaches coming almost every year[14, 11]. One of it's advantages over wheel or inertial odometry methods is that it can be used on any vehicle (air, underwater, land), and costs relatively cheap sensors (as compared to high precision Inertial Measurement Units). One of its disadvantages compared to other methods is that it is computationally expensive, and can work only in well-lit areas which have sufficient texture to recognize and track feature points.

1.1 Why Monocular?

The monocular approach is still interesting because the stereo case degenerates to the monocular case when the baseline is too small as compared to the distances of landmarks from the camera.

1.2 Application

Visual Odometry was originally intended to be used on Mars Rover [5], where there is no GPS and wheel odometry becomes unreliable due to slip on the sandy martian surface. In recent years, visual odometry has also found uses

in autonomous driving, wearable electronics, augmented reality, and even gaming [1].

1.3 Related Work

A detailed review on the progress of Visual Odometry can be found on this two-part tutorial series[6, 10]. Work on visual odometry was started by Moravec[12] in the 1980s, in which he used a single sliding camera to estimate the motion of a robot rover in an indoor environment. It was a stereo approach (the sliding camera took one photo at its original position and another on sliding to its other position). A separate feature detector, called the Moravec edge detector was developed to track points over several frames.
Most of the early stereo approaches (before 2004) involved triangulating the 3D points from the stereo pair in successive frames and then aligning them using least squares minimisation or the iterative closes point algorithm. This was however completely changed by Nister [7] who used a 3D to 2D minimisation, that was well-suited to deals with the anisotropic errors seen in 3D triangulation (errors in depth direction are more than those in the other two dimensions.
Interest in monocular approaches started increasing with the rise in popularity of Micro Aerial Vehicles (MAVs), and several computationally efficient algorithms have been developed specifically for MAVs like SVO[9].

2 The Algorithm

2.1 Problem Formulation

Input We have a stream of gray scale images coming from a camera. Let the frames, captured at time t and t+1 be referred to as I^t, $I^(t+1)$. We have prior knowledge of all the intrinsic parameters, obtained via any one of the numerous calibration toolboxes available[2] along with a chessboard.

Output For every pair of images, we need to find the rotation matrix R and the translation vector t, which describes the motion of the vehicle between the two frames. The vector, t can only be computed upto a scale factor in our monocular scheme.

2.2 Algorithm Outline

1. Capture images: I^t, I^{t+1},
2. Undistort the above images.
3. Use FAST algorithm to detect features in I^t, and track those features to I^{t+1}. A new detection is triggered if the number of features drop below a certain threshold.
4. Use Nister's 5-point alogirthm with RANSAC to compute the essential matrix.
5. Estimate R, t from the essential matrix that was computed in the previous step.
6. Take scale information from some external source (like a speedometer), and concatenate the translation vectors, and rotation matrices.

2.3 Undistortion

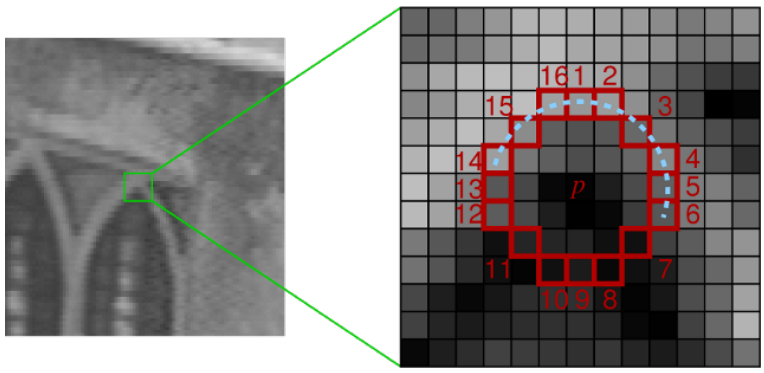
Distortion happens when lines that are straight in the real world become curved in the images. This step compensates for this lens distortion. It is performed with the help of the distortion parameters that were obtained during calibration.

2.4 Feature Detection

Our approach uses the FAST[8] corner detector. Suppose there is a point **P** which we want to test if it is a corner or not. We draw a circle of 16px circumference around this point as shown in 1. For every pixel which lies on the circumference of this circle, we see if there exits a continuous set of pixels whose intensity exceed the intensity of the original pixel by a certain factor **I** and for another set of contiguous pixels if the intensity is less by at least the same factor **I**. If yes, then we mark this point as a corner. A heuristic for rejecting the vast majority of non-corners is used, in which the pixel at 1,9,5,13 are examined first, and atleast three of them must have a higher intensity be amount at least **I**, or must have an intensity lower by the same amount **I** for the point to be a corner. This particular approach

is selected due to its computational efficiency as compared to other popular interest point detectors such as SIFT.

Figure 1: Image taken from original FAST paper[8]



2.5 Feature Tracking

Kanade-Lucas-Tomasi[3] feature tracker is used for finding sparse pixel wise correspondences. The KLT algorithm assumes that a point in the nearby space, and uses image gradients to find the best possible motion of the feature point. If, during the tracking procedure, the number of feature points go below 2000, then a new detection is triggered.

2.6 Essential Matrix Estimation

Once we have point-correspondences, we have several techniques for the computation of an essential matrix. The essential matrix is defined as follows:

$$y_1^T E y_2 = 0 \tag{1}$$

Here, y_1, y_2 are homogenous normalised image coordinates. While a simple algorithm requiring eight point correspondences exists[13], a more recent approach that is shown to give better results is the five point algorithm [4]. It solves a number of non-linear equations, and requires the minimum number of points possible, since the Essential Matrix has only five degrees of freedom.

RANSAC If all of our point correspondences were perfect, then we would have need only five feature correspondences between two successive frames to estimate motion accurately. However, the feature tracking algorithms are not perfect, and therefore we have several erroneous correspondence. A standard technique of handling outliers when doing model estimation is RANSAC. It is an iterative algorithm. At every iteration, it randomly samples five points from out set of correspondences, estimates the Essential Matrix, and then checks if the other points are inliers when using this essential matrix. The algorithm terminates after a fixed number of iterations, and the Essential matrix with which the maximum number of points agree, is used.

2.7 Computing R, t from the Essential Matrix

Another definition of the Essential Matrix (consistent) with the definition mentioned earlier is as follows:

$$E = R[t]_x \tag{2}$$

Here, R is the rotation matrix, while $[t]_x$ is the matrix representation of a cross product with t. Taking the SVD of the essential matrix, and then exploiting the constraints on the rotation matrix, we get the following:

$$E = U\Sigma V^T \tag{3}$$

$$[t]_x = VW\Sigma V^T \tag{4}$$

$$R = UW^{-1}V^T \tag{5}$$

2.8 Constructing Trajectory

Let the pose of the camera be denoted by R_{pos}, t_{pos}. We can then track the trajectory using the following equation:

$$R_{pos} = RR_{pos} \tag{6}$$

$$t_{pos} = t_{pos} + tR_{pos} \tag{7}$$

Note that the scale information of the translation vector t has to be obtained from some other source before concatenating.

2.9 Heuristics

Most Computer Vision algorithms are not complete without a few heuristics thrown in, and Visual Odometry is not an exception.

Dominant Motion is Forward The entire visual odometry algorithm makes the assumption that most of the points in its environment are rigid. However, if we are in a scenario where the vehicle is at a stand still, and a buss passes by (on a road intersection, for example), it would lead the algorithm to believe that the car has moved sideways, which is physically impossible. As a result, if we ever find the translation is dominant in a direction other than forward, we simply ignore that motion.

3 Results and Conclusions

A monocular algorithm for Visual Odometry was presented, and results on the KITTI benchmark have been presented here. The graphs presented in this report are for the results on one of the most challenging sequence of the KITTI Visual Odometry benchmark (sequence 00). It was observed that the Visual Odometry estimate was as good as the ground truth for upto 1000 frames**??**. Things start drifting off slightly from there, and due to the Visual odometry being a dead reckoning method, there is no way to correct these errors. However, the results remained within reasonable error bounds.
As far as the computational performance is concerned, the algorithms written in C++/OpenCV runs at 4.3 FPS on a single core of Intel i7.

4 Future Work

It is intended that this work would be used for Visual Odometry on fixed wing aircrafts like the Cessna - 206. Visual Data has been collected using a GoPro3 Black Edition, using a custom-made Cessna mount. There are challenges being encountered in getting reliable ground truth data.Also, the fish-eye lens of the GoPro introduces significant distortion, removal of which returns only a cropped version of the original image, thus reducing the resolution, and thus the scope for tracking a large number of feature point.

Figure 2: 1000 Frames

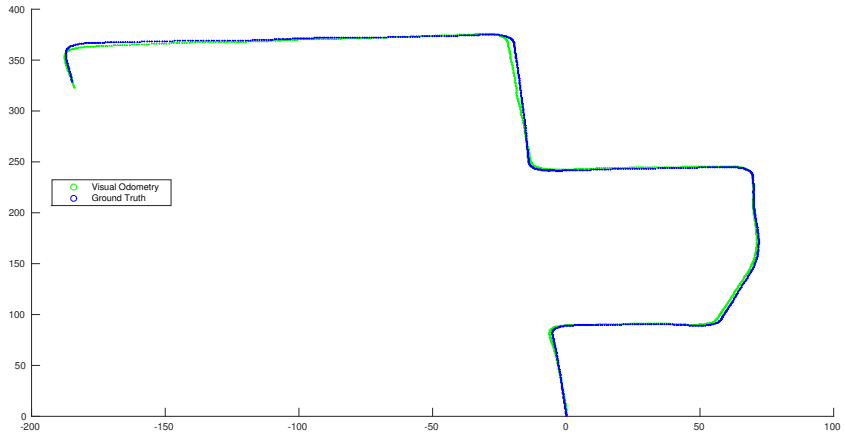


Figure 3: 2000 Frames

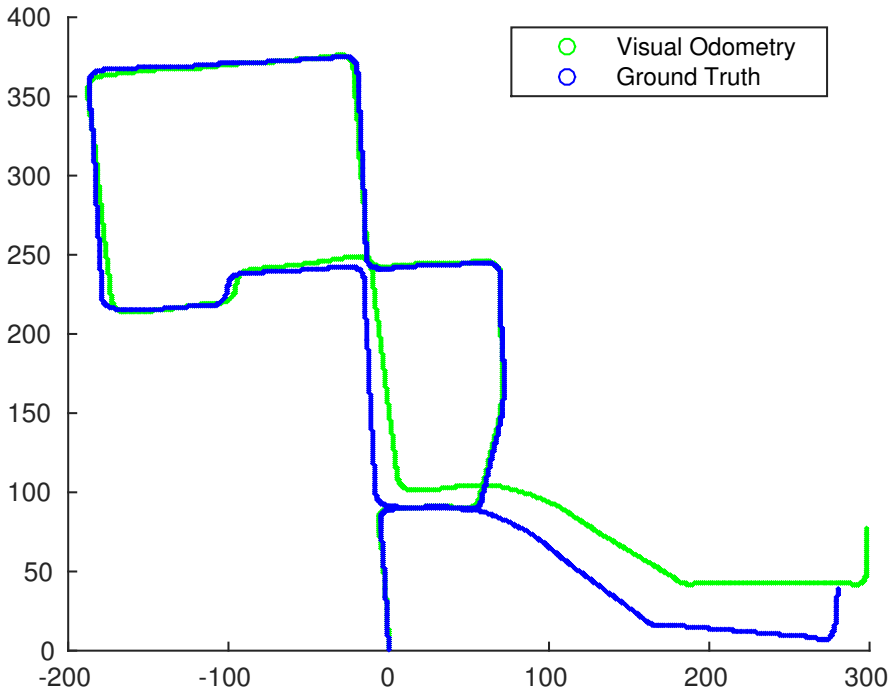
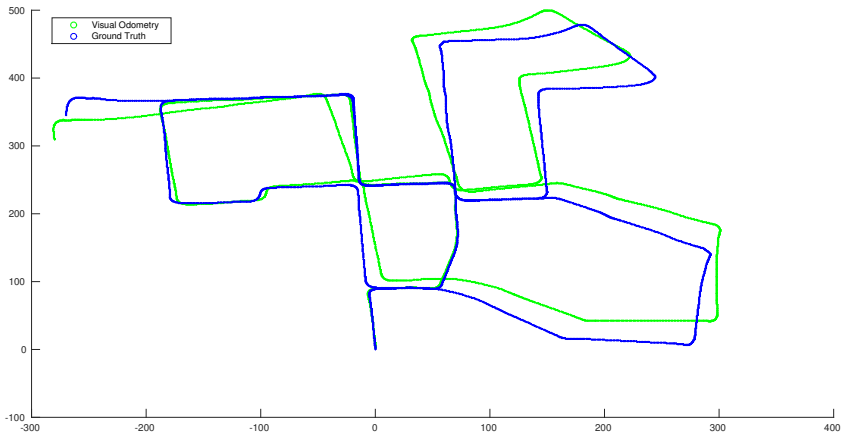


Figure 4: 4000 Frames



5 References

[1] https://www.google.com/atap/projecttango/.

[2] http://www.mathworks.in/help/vision/examples/stereo-calibration-and-scene-reconstruction.html.

[3] Takeo Kanade Carlo Tomasi. Detection and tracking of point features. In *CMU Tech Report*, 1991.

[4] Takeo Kanade Carlo Tomasi. An efficient solution to the five-point relative pose problem. In *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, 2004.

[5] Yang Cheng, Mark Maimone, and Larry Matthies. Visual odometry on the mars exploration rovers. In *SMC05*, 2005.

[6] Friedrich Fraundorfer Davide Scaramuzza. Visual Odometry: Part 1. *IEEE Robotics and Automation Magazine*, 2011.

[7] J.Bergen D.Nister, O.Naroditsky. Visual odometry. In *IEEE International Conference on Computer Vision and Patter Recognition*, 2004.

[8] Tom Drummond Edward Rosten. Machine learning for high-speed corner detection. In *European Conference on Computer Vision*, 2006.

[9] Christian Forster, Matia Pizzoli, and Davide Scaramuzza. SVO: Fast semi-direct monocular visual odometry. In *IEEE International Conference on Robotics and Automation (ICRA)*, 2014.

[10] Davide Scaramuzza Friedrich Fraundorfer. Visual Odometry: Part 2. *IEEE Robotics and Automation Magazine*, 2012.

[11] Takeo Kanade Herman Badino, Akihiro Yamamomto. Visual odometry by mutli-frame integration. In *International Workshop on Computer Vision for Autonomous Driving*, 2014.

[12] H.Moravec. *Obsacle Avoidance and Navigation in the Real worlf by a seeing robot rover*. PhD thesis, Stanford University, 1980.

[13] H.C. Longuet-Higgins. A computer algorithm for reconstructing a scene from two projections. In *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, 2004.

[14] Guangming Xiong Davide Scaramuzza Yanhua Jiang, Huiyan Chen. Icp stereo visual odometry for wheeled vehicles based on a 1dof motion prior. In *IEEE International Conference on Robotics and Automation*, 2014.