DEPARTMENT OF ELECTRICAL ENGINEERING, IIT-KANPUR

A Stereo Visual Odometry Implementation

Avi Singh, 12177
Under the Guidance of Prof. KS Venkatesh
May 25, 2015

1 ABSTRACT

This report presents a MATLAB-based implementation of a stereo Visual Odometry algorithm[13].It takes input from a calibrated stereo rig, and incrementally computes the 6-DOF pose of an agent moving through an environment. The algorithm is run on the KITTI Vision Benchmark[4].

2 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[15, 10]. 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.

2.1 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].

2.2 Related Work

A detailed review on the progress of Visual Odometry can be found on this two-part tutorial series[6, 9]. 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.

3 The Algorithm

3.1 Problem Formulation

Input We have a stream of gray scale images coming from a pair of cameras. Let the left and right frames, captured at time t and t+1 be referred to as I_l^t, I_r^t, I_l^{t+1}, I_r^{t+1}. We have prior knowledge of all the intrinsic as well as extrinsic calibration parameters of the stereo rig, obtained via any one of the numerous stereo calibration toolboxes available[2] along with a chessboard.

Output For every pair of stereo 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.

3.2 Algorithm Outline

1. Capture images: I_l^t, I_r^t, I_l^{t+1}, I_r^{t+1}
2. Undistort, Rectify the above images.
3. Compute the disparity map D^t from I_l^t, I_r^t and the map D^{t+1} from I_l^{t+1}, I_r^{t+1}.
4. Use FAST algorithm to detect features in I_l^t, I_l^{t+1} and match them.
5. Use the disparity maps D^t, D^{t+1} to calculate the 3D posistions of the features detected in the previous steps. Two point Clouds $\mathscr{W}^t$, $\mathscr{W}^{t+1}$ will be obtained
6. Select a subset of points from the above point cloud such that all the matches are mutually compatible.
7. Estimate R, t from the inliers that were detected in the previous step.

3.3 Undistortion, Rectification

Before computing the disparity maps, we must perform a number of preprocessing steps.

Undistortion This step compensates for lens distortion. It is performed with the help of the distortion parameters that were obtained during calibration.

Rectification This step is performed so as to ease up the problem of disparity map computation. After this step, all the epipolar lines become parallel to the horizontal, and the disparity computation step needs to perform its search for matching blocks only in one direction.
The images that we use in this implementation are from the KITTI Vision Benchmark Suite, and have already been undistorted and rectified, so we skip over the details of these two steps.

Figure 3.1: All the epipolar lines have become parallel to the x-axis in the image plane

3.4 Disparity Map Computation

Disparity Map Given a pair of images from a stereo camera, we can compute a disparity map. Suppose a particular 3D in the physical world F is located at the position (x, y) in the left image, and the same feature is located on $(x + d, y)$ in the second image, then the location (x, y) on the disparity map holds the value d. Note that the y-cordinates are the same since the images have been rectified. Thus, we can define disparity at each point in the image plane as:

$$d = x_l - x_r \tag{3.1}$$

Block-Matching Algorithm Disparity at each point is computed using a sliding window. For every pixel in the left image a 15x15 pixels wide window is generated around it, and the

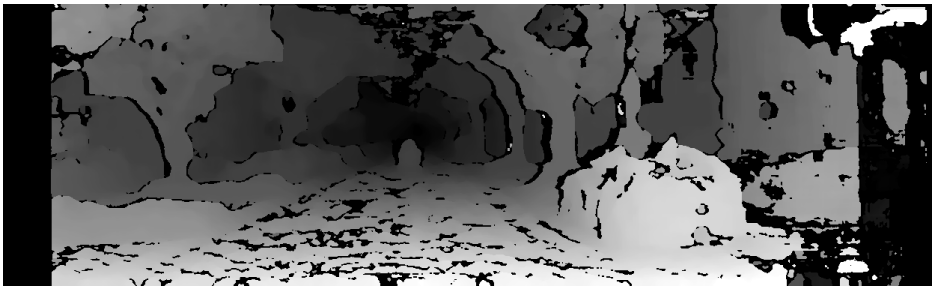


Figure 3.2: Disparity map computed from a stereo pair from the KITTI dataset

value of all the pixels in the windows is stored. This window is then constructed at the same coordinate in the right image, and is slid horizontally, until the Sum-of-Absolute-Differences (SAD) is minimized. The algorithm used in our implementation is an advanced version of this block-matching technique, called the Semi-Global Block Matching algorithm[11].A function directly implements this technique in MATLAB.

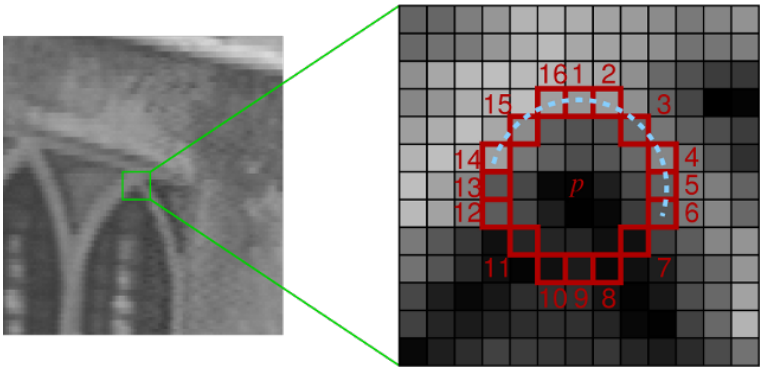
3.5 Feature Detection

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 3.3. 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.

3.6 Feature Description and Matching

The fast corners detected in the previous step are fed to the next step, which uses a BRISK[14] descriptor (a binary feature vector). The corners detected in I_l^t are matched to the corners detected in I_l^{t+1} Let the set of features detected in I_l^t be $\mathscr{F}^t$, and the set of corresponding features in I_l^{t+1} be $\mathscr{F}^{t+1}$.

Figure 3.3: Image taken from original FAST paper[8]



3.7 Triangulation of 3D PointCloud

The real world 3D coordinates of all the point in $\mathscr{F}^t$ and $\mathscr{F}^{t+1}$ are computed with respect to the left camera using the disparity value corresponding to these features from the disparity map, and the known projection matrices of the two cameras $\mathbf{P}_1$ and $\mathbf{P}_2$. We first form the reprojection matrix **Q**, using data from **P1** and **P2**:

$$Q = \begin{bmatrix} 1 & 0 & 0 & -c_x \\ 0 & 1 & 0 & -c_y \\ 0 & 0 & 0 & -f \\ 0 & 0 & -1/T_x & 0 \end{bmatrix}$$

c_x = x-coordinate of the optical center of the left camera (in pixels)
c_y = y-coordinate of the optical center of the left camera (in pixels)
f = focal length of the first camera
T_x = The x-coordinate of the right camera with respect to the first camera (in meters)

We use the following relation to obtain the 3D coordinates of every feature in $\mathscr{F}_l^t$ and $\mathscr{F}_l^{t+1}$:

$$\begin{bmatrix} X \\ Y \\ Z \\ 1 \end{bmatrix} = \mathbf{Q} \times \begin{bmatrix} x \\ y \\ d \\ 1 \end{bmatrix} \tag{3.2}$$

Let the set of point clouds obtained from be referred to as $\mathscr{W}^t$ and $\mathscr{W}^{t+1}$.

3.8 The Inlier Detection Step

This algorithm defers from most other visual odometry algorithms in the sense that it does not have an outlier detection step, but it has an inlier detection step. We assume that the scene is rigid, and hence it must not change between the time instance t and $t+1$. As a result, the distance between any two features in the point cloud $\mathscr{W}^t$ must be same as the distance between the corresponding points in $\mathscr{W}^{t+1}$. If any such distance is not same, then either there

Figure 3.4: Matching features on a over-layed false color image

is an error in 3D triangulation of at least one of the two features, or we have triangulated a moving, which we cannot use in the next step. In order to have the maximum set of consistent matches, we form the consistency matrix **M** such that:

$$\mathbf{M}_{i,j} = \begin{cases} 1, & \text{if the distance between i and j points is same in both the point clouds} \\ 0, & \text{otherwise} \end{cases} \tag{3.3}$$

From the original point clouds, we now wish to select the largest subset such that they are all the points in this subset are consistent with each other (every element in the reduced consistency matrix is 1). This problem is equivalent to the Maximum Clique Problem[3], with **M** as an adjacency matrix. A cliques is basically a subset of a graph, that only contains nodes that are all connected to each other. An easy way to visualise this is to think of a graph as a social network, and then trying to find the largest group of people who all know each other. This problem is known to be NP-complete, and thus an optimal solution cannot be found for any practical situation. We therefore employ a greedy heuristic that gives us a clique which is close to the optimal solution:

1. Select the node with the maximum degree, and initialize the clique to contain this node.
2. From the existing clique, determine the subset of nodes v which are connected to all the nodes present in the clique.
3. From the set v, select a node which is connected to the maximum number of other nodes in v. Repeat from step 2 till no more nodes can be added to the clique.

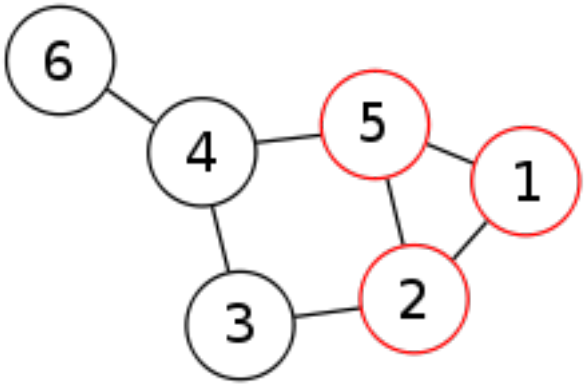


Figure 3.5: The largest clique in the above graph contains 1,5,2

3.9 Computation of **R** and **t**

In order to determine the rotation matrix **R** and translation vector **t**, we use Levenberg-Marquardt non-linear least squares minimization to minimize the following sum:

$$\epsilon = \sum_{\mathscr{F}^t, \mathscr{F}^{t+1}} (\mathbf{j_t} - \mathbf{PTw_{t+1}})^2 + (\mathbf{j_{t+1}} - \mathbf{PT^{-1}w_t})^2 \tag{3.4}$$

$\mathscr{F}^t, \mathscr{F}^{t+1}$: Features in the left image at time t and $t+1$ $\mathbf{j_t}, \mathbf{j_{t+1}}$: 2D Homogeneous coordinates of the features $\mathscr{F}^t, \mathscr{F}^{t+1}$
$\mathbf{w_t}, \mathbf{w_{t+1}}$: 3D Homogeneous coordinates of the features $\mathscr{F}^t, \mathscr{F}^{t+1}$
P: 3×4 Projection matrix of left camera
T: 4×4 Homogeneous Transformation matrix

The Optimization Toolbox in MATLAB directly implements the Levenberg-Marquardt algorithm in the function lsqnonlin, which needs to be supplied with a vector objective function that needs to be minimized, and a set of parameters that can be varied.

3.10 Validation of results

A particular set of **R** and **t** is said to be valid if it satisfies the following conditions:

1. If the number of features in the clique is at least 8.
2. The reprojection error ϵ is less than a certain threshold.

The above constraints help in dealing with noisy data.

4 Results

The algorithm is tested on the KITTI[4] vision benchmark suite. This section will be updated soon.

REFERENCES

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

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

[3] *Handbook of Combinatorial Optimization*, chapter The Maximum Clique Problem. Kluwer Academic Publishers, 1999.

[4] Raquel Urtasun Andreas Geiger, Philip Lenz. Are we ready for autonomous driving? the kitti vision benchmark suite. In *IEEE Conference on Computer Vision and Pattern Recongnition (CVPR)*, 2012.

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

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

[11] H Hirschmuller. Accurate and efficient stereo processing by semi-global matching and mutual information. In *International Conference on Computer Vision and Pattern Recognition*, 2005.

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

[13] Andrew Howard. Real-time stereo visual odometry for autonomous ground vehicles. In *IEEE International Conference on Intelligent Robots and Systems*, 2008.

[14] Margarita Chli Stefan Leutenegger and Roland Y. Siegwart. Brisk: Binary robust invariant scalable keypoints. In *IEEE International Conference on Computer Vision*, 2006.

[15] 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.