UNIVERSITY OF MARYLAND, COLLEGE PARK

Visual Odometry

Pranav Inani

pranavinani94@gmail.com

This report is a commentary on the steps involved in systematically performing Visual Odometry for Self-Driving Cars. We begin by performing some image processing techniques like demosaic and undistort to initially prepare each frame. Then, the Linear Algebra is used to find the fundamental matrix. Next, the essential matrix is extracted using the intrinsic matrix. Then the valid rotation and translations are obtained using 3-d reconstruction. The software implementation implementation on MATLAB®[1] is discussed wherever deemed necessary. Finally, the results and conclusions of the project will be stated. The report will then conclude by touching upon possible future work. It should be noted that the dataset was obtained from Oxford University.

Keywords: Visual Odometry, Self-driving Cars, 3-D Reconstruction, 8-point Algorithm.

CONTENTS

1 METHODOLOGY

1.1 PREPARE IMAGES

First, the intrinsic camera matrix was extracted. Then, the demosaic function was used to extract RGB image from the Bayer format for each image. Then, the undistort function was used to remove any distortion in the image. Finally, the images were converted into gray-scale in order to be compatible with feature extractors.

1.2 EXTRACT FEATURES AND MATCH POINTS

Next, we take two images and extract SURF features for each image. Then we use the MATLAB function to extract feature vectors, also known as descriptors, and their corresponding locations. The 'Upright' argument is set to true since we are not interested in the rotation information from the function.

Now, we use the match features function to find point correspondences for each pair op images. The 'Unique' flag is set to true since we want the correspondences to be unique. The 'MaxRatio' is reduced to 0.3 (default: 0.6) in order to get fewer and stronger correspondences.

1.3 ESTIMATE FUNDAMENTAL MATRIX

We use the 8-point algorithm [2] to estimate the fundamental matrix. Before we begin, perform some preprocessing. We start by scaling and translating the image points so that the centroid of the points is at the origin, and the average distance of the points to the origin is square root of 2. Next, we take any 8 point correspondences at random and construct the A matrix as prescribed by the algorithm. We then compute the 9 elements of the initial estimate of the fundamental matrix by taking the SVD of A. The least square solution is given by the last column of the right eigen vectors of A corresponding to the least singular value. Lastly, we obtain the final Fundamental matrix by enforcing the rank-2 condition by setting the last element of the diagonal matrix of its SVD to 0.

1.4 ESTIMATE ESSENTIAL MATRIX

From the fundamental, we use the intrinsic camera parameters matrix K to compute the initial estimate of the fundamental matrix. Then we enforce rank to constraint as before and also use the additional property that the non-zero eigen values of the Essential Matrix should be equal. We do this by taking the SVD of E and setting the diagonal matrix as diag($[1,1,0]$) and reconstructing the essential matrix.

1.5 Estimate Rotation and Translation

Once we have the Essential matrix we can create the 4 possible choices of rigid body transformations using the SVD of E and constructing the appropriate W matrix as

$$W = \begin{pmatrix} 0 & -1 & 0 \\ 1 & 0 & 0 \\ 0 & 0 & 1 \end{pmatrix}$$

The four choices are then of the form:
$[U * W * V' \quad U(:,3); \quad 0001]$
$[U * W * V' \quad -U(:,3); \quad 0001]$
$[U * W' * V' \quad U(:,3); \quad 0001]$
$[U * W' * V' \quad -U(:,3); \quad 0001]$

where U and V are the left and right eigen vectors of the essential matrix

Next, we ensure that the rotations we obtained are legal. We do this by finding the determinant of each of the choices and if it is negative we multiply the rotation matrix with a -1 to make it a legal rotation. Next, we check for the z components of the translations and discard any choice of H which has a negative component since that pose corresponds to the view which is behind the image plane.

1.6 Triangulation

To find the correct rotation and translation we will need to reconstruct the actual 3D position of one of the point correspondences. We use the matching image points p1,p2 to re-project onto the 3-D space to estimate the actual unknown 3-D point. We do this by constructing the required skew symmetric matrices and M matrices corresponding to both the points and find the point using the SVD.

If the z translations of both the point estimates are positive we accept the corresponding H matrix. Otherwise, the next set of point correspondences is tried. If all the point correspondences fail to give a suitable match, the identity matrix is passed in as rotation and a vector of zeros as translation.

1.7 Update Rotation and Translation

Initially, we set the the Rotation to a 3x3 Identity matrix and translation to the $[0,0,0]$ vector i.e., we assume that we are at the origin of the world and update our rotations and translations with respect to these coordinates.

We update the rotations and translations as:
$t = t + R * t_{new}$;
$R = R * R_{new}$;
Finally, we plot the x and z components of the translations to estimate the approximate path taken by the car.

2 Results, Discussions & Future Work

The above pipeline was used to generate the following results:

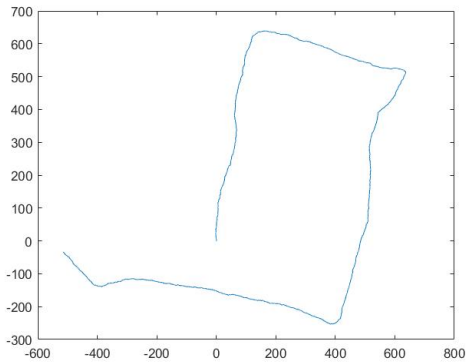

Figure 2.1: Example Output 1

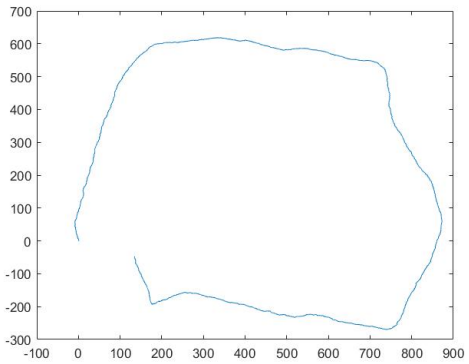

Figure 2.2: Example Output 2

Since the points are chosen at random the results may not be repeatable. However the figure resembles the approximately rectangular path taken by the vehicle.

The RANSAC technique could be used in the future to serve as a better way of choosing point correspondences rather than choosing any 8 point correspondences at random. Additionally, bundle adjustment maybe used to reduce reprojection errors to improve pose estimation. These are potential future works.

References

[1] MATLAB, *version 9.3.0 (R2017b)*. Natick, Massachusetts: The MathWorks Inc., 2017.

[2] R. I. Hartley, "In defense of the eight-point algorithm," *IEEE Transactions on pattern analysis and machine intelligence*, vol. 19, no. 6, pp. 580–593, 1997.