Project 2: Visual Odometry

Indushekhar Prasad Singh

ENPM 673 Perception for Autonomous Robots
Masters of Engineering Robotics

A. JAMES CLARK
SCHOOL OF ENGINEERING

University of Maryland , College Park
United State of America

1 Project Pipeline

1.1 Preprocessing

The input images are in Bayer format from which color images were recovered using the demosaic function with GBRG alignment. Then image frames were undistorted using the MATLAB function provided in the input. For applications such as Visual Odometry, it is important to know the real world location of points since the nonlinear nature of the lens distortion makes the problem challenging.

1.2 Extraction of Camera Parameters

Intrinsic matrix of the camera is calculated using the the MATLAB function ReadCameraModel.m. The function gives the values of focal length (f_x and f_y) and Principal point offset (c_x and c_y).

1.3 Feature detection and Correspondence

SURF feature detector was used for detecting features. Then descriptor or feature vectors were extracted using **extractFeatures** function of MATLAB computer vision toolbox. **matchFeatures** function was used to locate the point with matching features.

1.4 Estimation of Fundamental matrix

Estimating fundamental matrix is the key step involved in the whole pipeline.The 8-point normalized algorithm was implemented to obtain the fundamental matrix. Using the matching points, a RANSAC function was implemented to randomly sample 8 set of matching points and checking the score of the resulting matrix with reference to other matching points. A fitness score was used to update the value of fundamental matrix. The perpendicular distance of matching point from the epipolar line is computed . If the distance lies within the threshold selected we increment the score by 1. Thus, the fundamental matrix gets updated if its score is greater than the earlier score

1.5 Estimation of Essential matrix

Essential Matrix is computed after this by using the camera intrinsic and Fundamental matrix. It is made sure that the rank of the matrix is two by only keeping two singular values in the diagonal matrix of the svd. MATLAB code snippet is give below :

```
 %% Calculating the essential matrix
E = K'* F * K;
[U,~,V] = svd(E);
E = U * [1 0 0;0 1 0;0 0 0] * V';
E = E / norm(E);
```

1.6 Extracting camera pose

For a given essential matrix $E = Udiag(1, 1, 0)V^T$, and first camera matrix $P1 = [I|0]$, there are four possible choices for the second camera matrix P2 namely : $P2 = [UWV^T| + u3]$ or $[UWV^T| - u3]$ or $[UW^TV^T| + u3]$ or $[UW^TV^T| - u3]$
where W is orthogonal matrix equal to [0 -1 0;1 0 0;0 0 1];

1.7 Selection of correct pose

Now we need to select one correct pose out of 4 poses derived from previous steps. I have used the constrained which is specific for this project to estimate the correct pose :
• Assuming that car is moving forward only , we can filter out the transform which has positive z translation.
• **Rotation and Translation condition around y axis :** For the current scenario, rotation can be assumed to be only around y axis. So that we can filter out the rotation matrix whose rotation component along other axes is approximately zero. While selecting the correct pose, the pose with minimum y translation is also one of the condition utilized in computing the correct pose.

Finally, if none of the poses are able to satisfy all the conditions then pose of the world frame is selected as correct pose.

1.8 Camera pose update

The last step before getting the final camera pose is updating the pose with respect to previous frame. For updating the pose of first frame we use the pose of the world frame and then subsequent frames uses the pose of previous frames. Simple equation for that is given below :

$$R_{next} = R_{current} \times R_{next}$$

$$t_{next} = t_{current} + R_{current} \times t_{next}$$

2 Output

Output can be seen by running the MATLAB file **project2Main.m**. The Plot of the position of the camera center (for each frame) based on the rotation and translation parameters between successive frames thus obatained is given below:

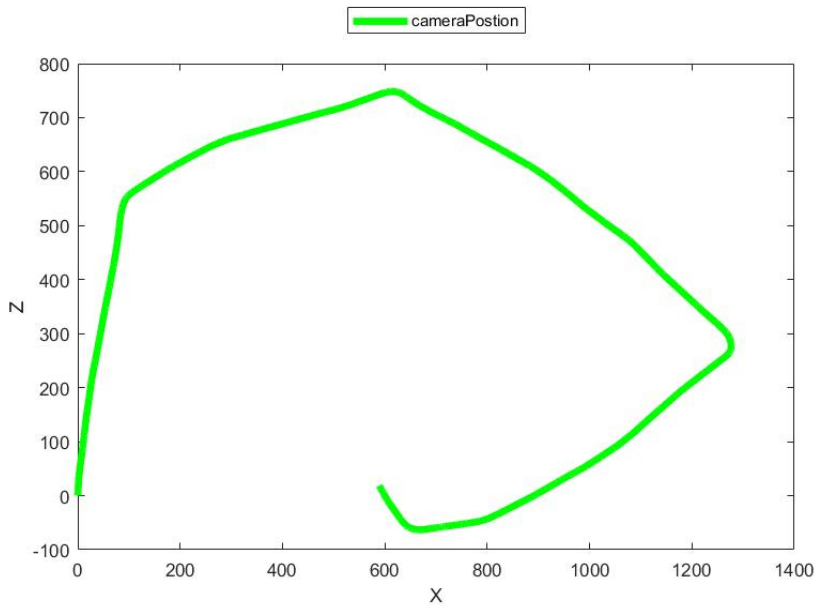
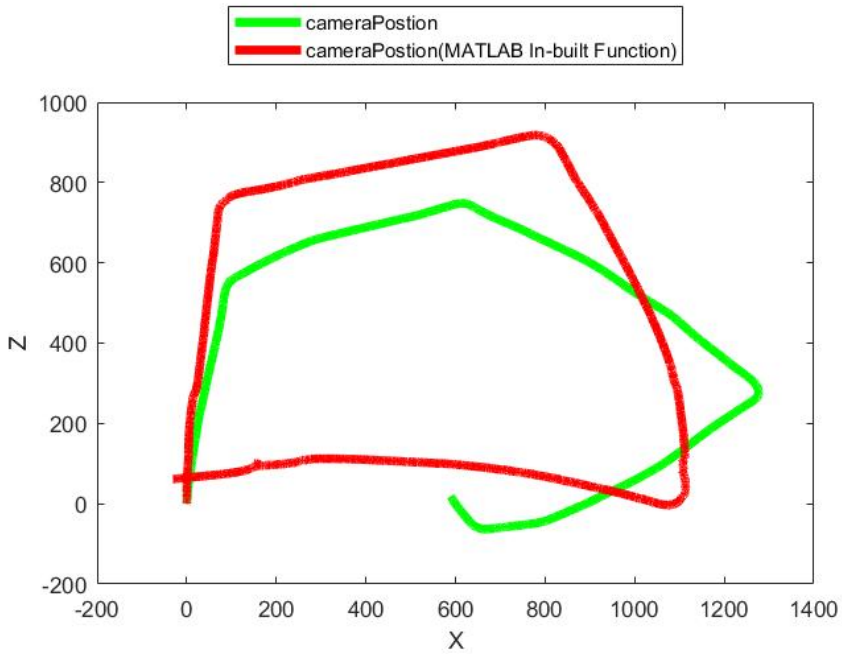


Fig 1. Plot of the camera position in x-z plane

World-Z direction is forward and world-X direction is towards the right. The camera position is changing as per car movement and all the hard right turn can be easily seen in the plot, though drift is there in the value of z axis.

3 Comparison with output from MATLAB functions

Comparison plot can be seen by running the code **project2MatlabComparison.m**. All the steps of the pipeline till feature detection is same in this case also. Fundamental matrix is calculated using the function **estimateFundamentalMatrix**, which also gives index of inliers points. Only these inlier points are used to calculate the realtive pose using **relativeCameraPose** function. The pose obtained is updated as previous to give the final camera pose.



Significant improvement can be seen in the plot obtained by using MATLAB functions. Turns are sharp and movement along z axis is also greater. The car also reaches the point it started (as it is the ground truth).

References

[1] https://www.robots.ox.ac.uk/ vgg/hzbook/hzbook2/HZepipolar.pdf

[2] http://www.mathworks.com/help/vision/index.html