# ENPM 673 – Robotics Perception

## Project 2 – Visual Odometry

## 1. Introduction

In this project we aim to perform Visual Odometry for estimating the trajectory of the Robot(camera).

## 2. Getting Started and Running the Code

The code is written in MATLAB Version R2018a – Full Version. The project consists of the file ***visualOdometry.m*** which is the main file in the **Code** folder. To execute the project, simply run the ***visualOdometry.m*** file in MATLAB.

Kindly copy the 'Oxford_dataset' folder to the **Input** Folder. Workspace file '**workspaceFile'** is present in the Output Folder.

## 3. Visual Odometry Pipeline

### 3.1 Image Pre-processing

The input images are in the Bayer format which are converted to RGB image using the demosaic function.

Fig 1. Bayer Image

Fig 2. RGB Image

After demosaicing of the image, the distorted Image is undistorted which will then be used for feature detection.

Fig 3. Undistort

### 3.2 Feature Point Detection

In this step, the image is searched for salient key points that are likely to match well in other images. SURF which is a Blob detector is used for detecting features. SURF features are Rotation, Scale and Affine Invariant. They are faster than SIFT and are good at repeatability which are important when tracking features.


Fig 4. SURF features

### 3.3 Fundamental Matrix Computation

A fundamental Matrix is a 3x3 matrix that relates corresponding points in two images. Here, SURF features detected in subsequent images are used to estimate the Fundamental Matrix. Now, we encounter many outliers in the tracked features which give erroneous output. Therefore, RANSAC (Random Sample Consensus) is used for outlier rejection. This gives us a robust fundamental Matrix.

The function prototype is:

```
[Fbest, fMatrixInbuilt, inliersIndex] = estFundamentalMatrix([currentX
currentY] ,[nextX nextY], matchedPointsCurrent, matchedPointsNext, 0.001);
```

Function Name: `estFundamentalMatrix`

Output: `Fbest` - Custom Fundamental Matrix Computation
`fMatrixInbuilt` – Using Inbuilt Function
`inliersIndex` – Index for Inliers

### 3.4 Essential Matrix

The Essential Matrix is computed from the Fundamental Matrix and the Camera Intrinsic Parameters by Singular Value Decomposition. Rank 2 is enforced.

### 3.4 Camera Pose Estimation

This step gives us the possible rotation and translation matrices for a given essential matrix. We observe that we get 4 possible combinations of Rotations and Translation. This is stored in Tsolutions and Rsoutions.

Function Prototype:

```
[Tsolutions, Rsolutions, translateVector1, translateVector2, R1, R2] =
getCameraPose(E);
```

Function Name: `getCameraPose`

Now we must extract the correct camera pose from the 4 possible options. This is done through triangulation. In this method, we reconstruct the 3D point and check is it is in front of both the cameras. We will only get one solution for which this holds true. That is out Rotation and Translation.

```
[Tactual, Ractual] = getCorrectPose(Tsolutions, Rsolutions,
indexWithMetric(1:8,:), K);
```

Function Name: `getCorrectPose`

### 3.5 Output

Once we get translation and rotations for pair of frames, we update the global translation and rotation values. This is plotted on every iteration to get a graph that represents the motion of the vehicle. The full output(path) can be seen in the next image.

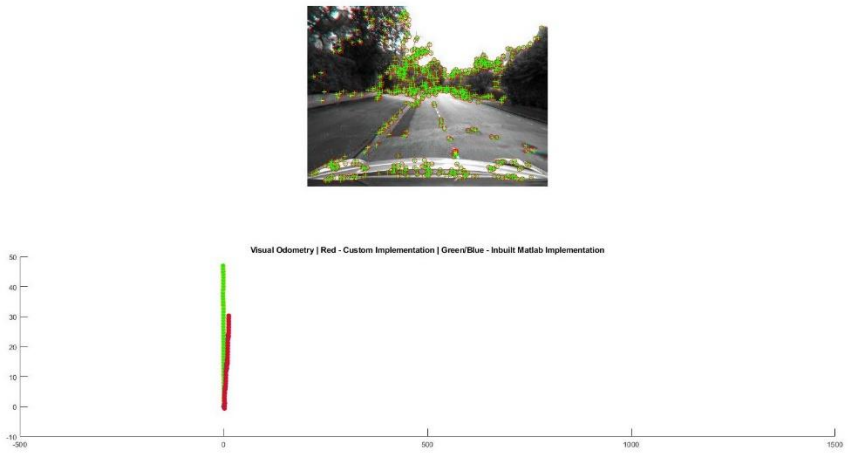


Fig 5. Features and Translation plot

**Extra Credit:**

Here we can see the plot which compares custom written functions (Red output) versus inbuilt functions of MATLAB (blue output). Although the overall structure appears to be the same, it is a scaled down version of the inbuilt implementation. We have no Loop-Closure and local adjustment (Bundle Adjustment) implementation which corrects the trajectory based on 'm' past values and closes a loop is the scene has been visited before. All these methods will help increase accuracy of the system.

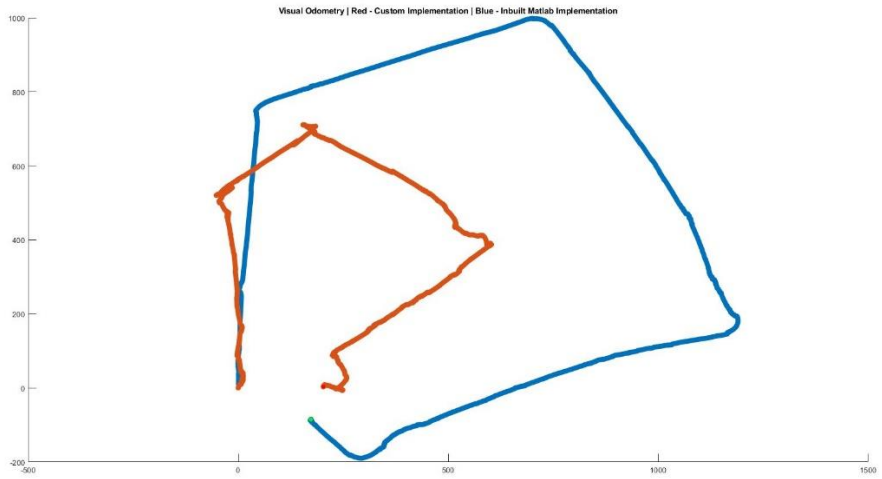


Fig 9. Plot Comparison

## 4. Acknowledgements

Implementation of the RANSAC function and Sampson distance for fundamental matrix estimation (`ransacn.m` and `funddist.m`) has been used from (http://www.peterkovesi.com/matlabfns/Robust/ransac.m) by Peter Kovesi.

## 5. References

- MATLAB Documentation
- ENPM673 – Project2.pdf
- Visual Odometry Part 1 and Part 2 by Davide Scaramuzza and Friedrich Fraundorfer.

## 6. Author

Rohitkrishna Nambiar (UID – 115507944)