Project 2 | Monocular Visual Odometry

Dataset

The data was provided to students via a zipped directory of the images; the images make up a high framerate video of an autonomous Nissan Leaf, coined the Oxford RobotCar, traveling through central Oxford. The center camera data was extracted via a 1x Point Grey Bumblebee XB3 trinocular stereo camera; said images are formatted as 1280x960x3.

Figure 1. collage of provided frames; data is given as distorted and uncolored

Camera Setup

Using a provided MATLAB script, parameters of the intrinsic camera matrix can be extracted and utilized in future transformations. Additionally, parameters are set such that the trajectory is started from the origin (0, 0) in the x-z coordinate space.

Image Processing

The present frame and immediately next frame are demosaiced from a Bayer encoded image into a true color image. Subsequently, another script is used to undistort the image using a provided Look-up Table (LUT). For the purpose of this assignment, the image is converted into the gray color space.

Figure 2. Pipeline for processing: Demosaic with 'gbrg' followed by undistortion

Corresponding Points

SURF features are detected in each of the two images using MATLAB's *detectSURFFeatures()* command. The output from this command is fed into *extractFeatures()* to extract all feature vectors from the now binary image. Finally, using the matched points between the two images are generated using the *matchFeatures()* command.

Figure 3. 1117 matched points between two sequential images, plotted

User-based Approach

Fundamental Matrix

A Random Sampling and Consensus (RANSAC) algorithm was written for the purpose of this project. The reason for selecting this particular fitting technique is its resistance to effects due to outliers in the data. The algorithm is run for 2000 iterations for every image such that inlier points fall below a selected error threshold of 1E-4. Additionally, due to massive generation of inlier points the code is written to simply return 10% of inliers to improve computation time. This RANSAC implementation utilizes another function written to generate a normalized fundamental matrix.

Essential Matrix

Using the output from the RANSAC Fundamental Matrix function, an Essential Matrix can be calculated by the following formula, assuming the provided intrinsic matrix (K) is lower triangular:

$$Essential\ Matrix = K * Fundamental\ Matrix * K^T$$

This function generated four possible solutions for the matrix [R|T] however, only one could be used for correct results.

Solution

A triangulation algorithm was written (adapted from Harley/Strum) to decide which of the four solutions is appropriate. From a high level, the correct [R|T] matrix would result in the camera traversing into the positive depth. As the triangulation iterates over the number of inlier point, the decision to limit total inliers to 10% of total improved computation time.

MATLAB-based Approach

Fundamental Matrix

The Fundamental Matrix is generated by calling MATLAB's *estimateFundamentalMatrix()* and passing arguments specifying: RANSAC method, 2000 iterations, and 1E-4 distance threshold. In addition to the fundamental matrix, an inlier index is returned to specify inlier points.

Solution

Camera pose is calculated by calling MATLAB's *relativeCameraPose()* command. By passing values for the camera's intrinsic matrix, matched inliers for both images, and the fundamental matrix, an internal triangulation can be conducted to generate an appropriate Rotation matrix and Translation vector.

Transformation

After outputting the index of the correct Rotation matrix and Translation vector, an [R|T] matrix can be formed and iteratively multiplied to itself. The values in T are then (iteratively) used as target values for camera position with respect to each new frame. The x and z values are extracted for trajectory plotting.

Results

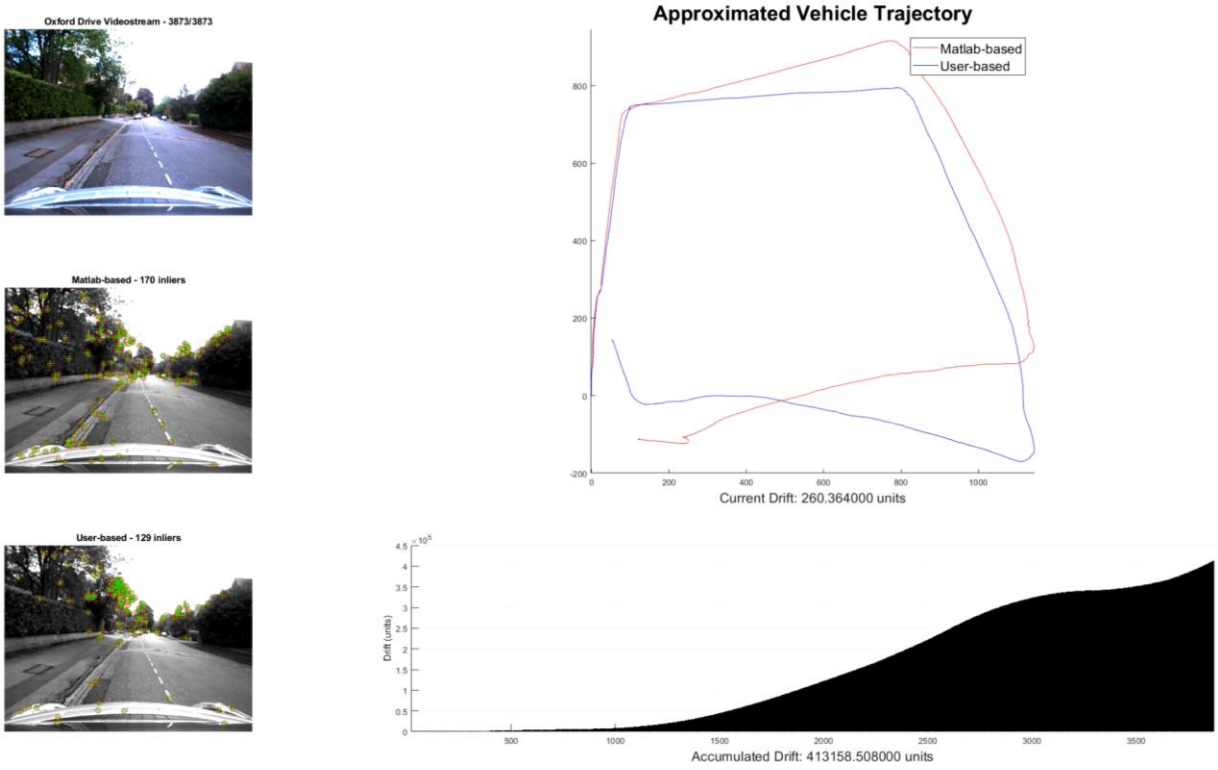

Figure 4. Figure layout for vehicle trajectory indicating: number of inlier points based on method, visualization of vehicle trajectory, and visual of drift accumulation between methods

The x-z camera coordinates are plotted in the Approximate Vehicle Trajectory window with red representing points generated by MATLAB's Computer Vision Toolbox and blue representing points generated by function developed over the course of this project. Although a sound comparison of efficacy would involve comparison with ground truth data, off-the-cuff observations indicate that the user-based trajectory is fairly consistent with the frames. Additionally, it appears that user-based is resilient to moving vehicles in the field-of-view (FOV); MATLAB-based incorrectly registers the third and final turn due to a moving vehicle in the FOV.

Analysis Video

The video can be downloaded at the following link (700MB):

https://www.dropbox.com/s/f24y83k5uyy1yui/Oxford_Trajectory2.avi?dl=0

References

https://avisingh599.github.io/vision/visual-odometry-full/

https://perception.inrialpes.fr/Publications/1997/HS97/HartleySturm-cviu97.pdf