Final Project CMSC426

Andrew Lee
May 19, 2021

1 Estimate Rotations and Translation between frames

1.1 Compute Intrinsic Matrix

All that is needed is to use fx, fy, cx, cy, _, LUT = ReadCameraModel('./Oxford_dataset_reduced/model') and K = np.array([[fx, 0, cx],[0, fy, cy],[0, 0, 1]]) from the project specifications. K ended up being approximately

$$\begin{bmatrix} 964.82 & 0 & 643.78 \\ 0 & 964.82 & 484.40 \\ 0 & 0 & 1 \end{bmatrix}$$

1.2 Load and Demosiac images

The functions in the project specifications are used except for the optional UndistortImage.py. cv2.imread() reads the pixel data from an image and turns it into a matrix. cv2.cv2Color() recolors an image by altering the values of each pixel according to a conversion code. An example of a resulting image that is colored and has the Bayer filtering removed looks like this:

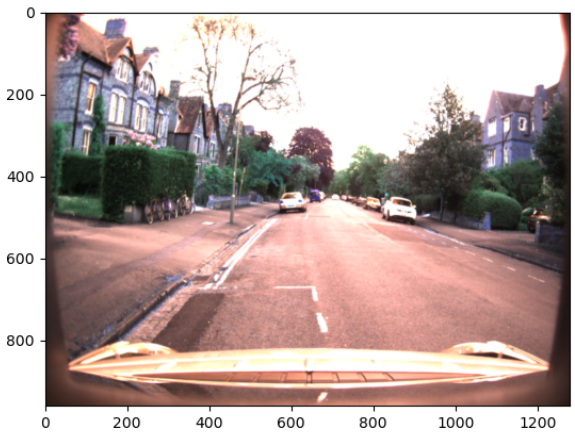

Figure 1: An image from the video, colorized and demosiaced

1.3 Keypoint Correspondences

I used brute force matching with knnMatch using SIFT descriptors. Mainly, I used the method from this website: https://www.docs.opencv.org/master/dc/dc3/tutorial_py_matcher.html. Although with a few alterations to fit the project. I first found the keypoints and descriptors of every image. cv2.SIFT_create() creates the feature matching algorithm object which we can then use detectAndCompute() to get a list of significant features of an image and their descriptors. After this, I used cv2.bfmatcher to create the brute force matching object with knnMatch to find related keypoints between 2 images. Then, to filter out unwanted matches, such as keypoints that have more than one match, I use the ratio test used in the website, however, I choose to store the coordinates of the matches instead of the matches themselves in order to use them for estimating the fundamental matrix.

1.4 Estimate the Fundamental Matrix

cv2.FindFundamentalMatrix() constructs the 3x3 fundamental matrix from 2 lists of corresponding points and a modeling algorithm of choice. For this project, I used RANSAC. The fundamental matrix represents the relation between the corresponding points in the images.

1.5 Recover Essential Matrix

Since we have the intrinsic matrix and the fundamental matrix, we can calculate the essential matrix with the equation

$$E = K'^T F K$$

However, because this is the same camera taking these pictures rapidly, I simplify $K' = K$ in the equation to make

$$E = K^T F K$$

The essential matrix is similar to the fundamental matrix, but it also accounts for the position and orientation of the camera.

1.6 Reconstruct Rotation and Transformation parameters from E

cv2.recoverPose takes in the essential matrix, corresponding points from the 2 images, and the intrinsic matrix to output the rotation and transformation matrices we need. It decomposes the camera matrix in one of four different ways to get the extrinsic parameters, which we can then take the rotation and translation matrices from.

2 Reconstruct the Trajectory

I used the method in the ELMS announcement to get the coordinates of the path of the camera. Simply put, we can find the point X_0^k by using the equation

$$X_0^k = inv(\begin{bmatrix} R_k & t_k \\ 0 & 1 \end{bmatrix} \begin{bmatrix} R_{k-1} & t_{k-1} \\ 0 & 1 \end{bmatrix} ... \begin{bmatrix} R_1 & t_1 \\ 0 & 1 \end{bmatrix}) \begin{bmatrix} 0 & 0 & 0 & 1 \end{bmatrix}$$

In the code, I use a method of having a cumulative matrix and multiplying in a new matrix for each new point I need to find, which saves having to recalculate the entire list of matrices from the beginning every time I want to find a new point. This will directly give all the points so that we can easily plot them. The results of which are shown below:

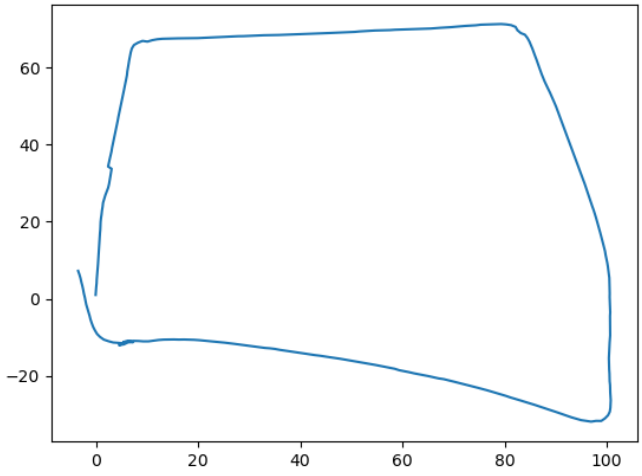

Figure 2: 2D representation of the pathof the car

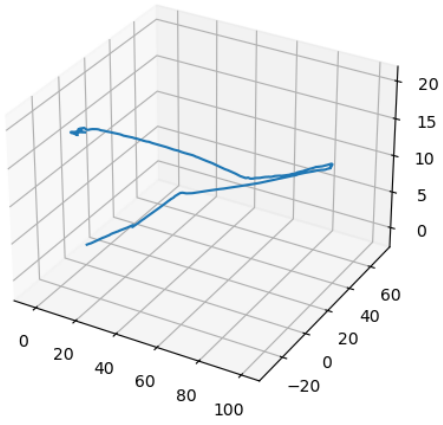

Figure 3: 3D representation of the path of the car