Adam Clay –U31446255

I decided to choose the track to implement the Odometry class without using cv2.findEssentialMat, and instead I created my own implementation to estimate the essential matrix. I first get keypoint features using the OpenCV ORB keypoint detector, then use the FlannBasedMatcher to match these keypoints between neighboring images. This is done for every pair of images. For the first image, I calculate the pose (position and orientation) simply using the identity matrix with a fourth column of 0's for computation: using this matrix the first coordinate will be all zeros. Next, I filter out the good matches using the distance between the found key points. I then save the good matches and pass the points to my implementation of RANSAC to calculate the best fundamental matrix using the eight point algorithm.

At first I tried using 8 random points, and the 8 points with the smallest distance, from the filtered matches, but I found that in most cases my estimated essential matrix (estimated by multiplying [dot product] the transpose of the camera calibration matrix with the estimated fundamental matrix, and then multiplying this result with the camera calibration matrix) was far off from the cv2 implementation that used RANSAC to estimate the essential matrix. To combat this, I implemented my own form of RANSAC. Finding hyperparameters that worked well for this task took a lot of trial and error, and I found, unfortunately, that those that worked best for the training video did not work so well on the test video. My algorithm works by iterating several hundred times (I found that 750 iterations was about as many as I could use to have my code finish in under 10 minutes, but if we could run our code for longer I would have been able to get better results with several thousand iterations) and choosing 8 random points in each iteration, calculating the fundamental matrix with my implementation of the normalized eight-point algorithm, and finding the errors that the current fundamental matrix would generate between images. I had to find a threshold that worked well for these errors that would result in an acceptable fundamental matrix. I found that .01 worked well on the training video, but resulted in near random results on the test video, so I ended up submitting my algorithm with a threshold of .1 which got me results well above the baseline. At the end of the iterations I have saved the fundamental matrix that results in the most inliers (within the threshold).

My normalized eight-point algorithm follows the implementation discussed in the slides very strictly: first, I center the key points and scale them accordingly, then pass these points through the regular eight-point algorithm, enforcing the rank-2 constraint, and finally use the means and scale values that I calculated before to calculate T and T-prime to renormalize the fundamental matrix to the original coordinates in the image. After getting the best fundamental matrix recovered from my RANSAC algorithm, I use cv2.RecoverPose to get the rotation matrix and translation vector. These are combined as shown in the slides to get the transformation matrix. The inverse of this transformation matrix, when multiplied with the previously calculated pose, will give the position and orientation of the next image. For each transformation I append the coordinates to the path, and finally return this accumulated path as my end result.

I found the main obstacle in this challenge was finding a way to estimate the essential matrix in a way comparable to the cv2 implementation, because I got fairly good results using the built-in function, but was more comfortable going on the track to estimate it myself. The hyperparameters to my RANSAC implementation took a long time to dial in in such a way that would result in good performance on the test video, and these ended up being different than those that worked well on the training video. I was also limited, more than I expected, by the time constraint because as many iterations in my RANSAC algorithm as possible, I found, resulted in the best estimation of the essential matrix which was a crucial estimation for solving this problem. Going forward, I would like to create a more generalizable implementation.