Sijia Hua
Uid: 116504805

tomhuasj@umd.edu

1 Estimate Rotations and Translations Between Frame

1.1 Compute Intrinsic Matrix

From the code provided in the pdf, the read camera model function can return several values. fx is the horizontal focal, fy is the vertical focal, cx is the horizontal principal point, and cy is the vertical principal point. To form the camera intrinsic matrix,

$$\begin{bmatrix} fx & 0 & cx \\ 0 & fy & cy \\ 0 & 0 & 1 \end{bmatrix}$$

By plugging those values, we can form the camera intrinsic matrix K.

1.2 Load and Demosaic Images

To load those files, I use a method[4] to load all the images in a folder. In each loop, after reading an image file, I use convert color method provided in the instruction to convert the image back to colored image and use undistorted img method to produce an undistorted image based on the undistortion lookup table provided by the read camera model function. After finishing all those steps, I put the image in the images list.

1.3 Keypoint Correspondences

To match keypoints, I choose to define a function outside the main loop so I can modify the code easily. For my feature matching, I choose to use flann matcher which is used in cv2 epipolar tutorial. [2][3]After matching the keypoints (with the same precision given in the tutorial), we need to find those best match points by calculating the distance. Instead of calculating manually in project 4, I choose to use the code from tutorial to calculate the distance, which gives us two set of points, pts1 and pts2. Both of them contains corresponding keypoints in two consecutive images.

1.4 Estimate Fundamental Matrix

After getting those best matching points, I need to process those points to get the fundamental matrix. cv2 provides a method called cv2.findfundamentalmat which can help us find the fundamental matrix by passing corresponding points. There are 4 methods available in this algorithm. I choose the RANSAC one since it produces a more accurate trajectory in the next section and it also uses over 8 points to calculate which also ensures accuracy. At the end of the method, I use mask to find inline points.[2]

1.5 Recover Essential Matrix

Since we have already calculated the camera intrinsic matrix K and fundamental matrix F, we can calculate the essential matrix easily using the formula. $F = K^{-T}EK^{-1}$ means that $E = K^TFK$

1.6 Reconstruct Rotation and Translation Parameters from E

In this part, I can calculate the rotation and translation easily by using the cv2.recoverPose, since we have already calculated the corresponding points, essential matrix, and the camera intrinsic matrix. From the documentation[1], the recoverpose method takes 4 inputs, E, pts1, pts2, and K. It will return 4 values, a return value, 3×3 rotation matrix, 3×1 translation matrix, with a mask. The only value we need is the rotation and translation matrix. After find the rotation and translation matrix for each pair of graphs, I stored them into two separate list.

2 Reconstruct the Trajectory

To reconstruct the trajectory, I choose to measure the camera center as the notes. We have already calculated the rotation and translation for each pair of graphs, now I need to process the value. I use numpy concatenate to form the matrix

$$\begin{bmatrix} R & t \\ 0 & 1 \end{bmatrix}$$

and get its inverse so that I can get the center of the camera in their coordinate system. For those consecutive graphs, I need to multiply the inverse of matrix to get those centers. By inverting the system, I get a series of homogeneous points. Since the last value is 1, and $\frac{x}{1} = 1$ $\frac{y}{1} = 1$ $\frac{z}{1} = 1$, we can use the x, y, z value directly. By using matplotlib in 3d projection, we get the following trajectory.

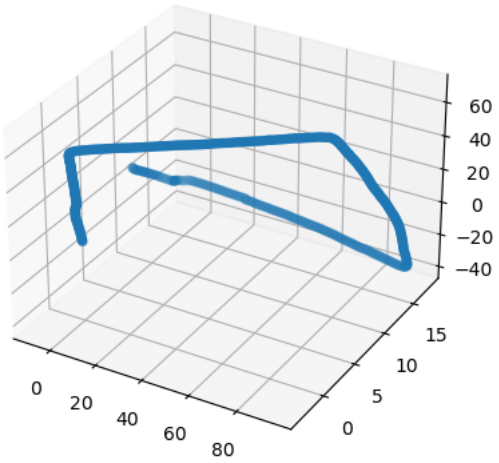

Figure 1: 3d trajectory

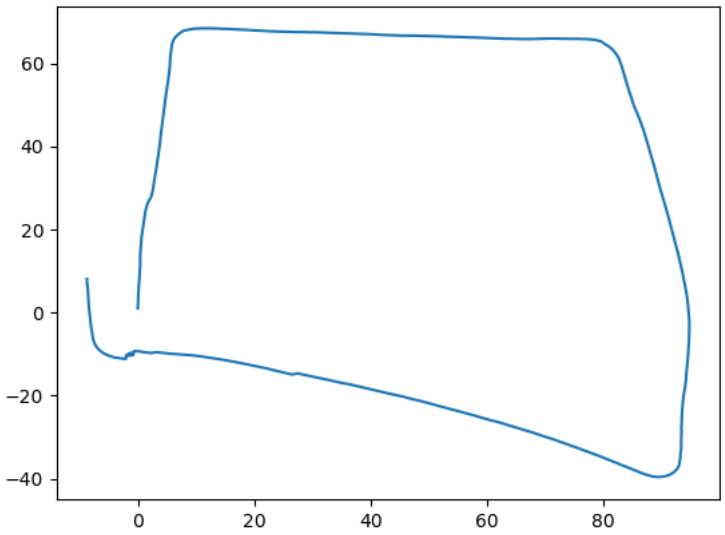

Figure 2: 2d trajectory

References

[1] *Camera Calibration and 3D Reconstruction*. URL: https://docs.opencv.org/3.4/d9/d0c/group__calib3d.html#gadb7d2dfcc184c1d2f496d8639f4371c0.

[2] *Epipolar Geometry*. URL: https://docs.opencv.org/master/da/de9/tutorial_py_epipolar_geometry.html.

[3] *Feature Matching*. URL: https://docs.opencv.org/master/dc/dc3/tutorial_py_matcher.html.

[4] *Loading all images using imread from a given folder*. URL: https://stackoverflow.com/questions/30230592/loading-all-images-using-imread-from-a-given-folder.