Project 5 Writeup

Instructions

- Provide an overview about how your project functions.
- Describe any interesting decisions you made to write your algorithm.
- Show and discuss the results of your algorithm.
- Feel free to include code snippets, images, and equations.
- List any extra credit implementation and result (optional).
- Use as many pages as you need, but err on the short side.
- **Please make this document anonymous.**

Project Overview

For this project I estimated the camera projection matrix, estimated the fundamental matrix, and implemented RANSAC to produce a point cloud of the object.

Implementation Detail

Part I: Camera Projection Matrix

I first constructed all the rows in the form $\begin{pmatrix} X & Y & Z & 1 & 0 & 0 & 0 & 0 & -Xu & -Yu & -Zu \end{pmatrix}$ then stacked them on top of all of the rows in the form $\begin{pmatrix} 0 & 0 & 0 & 0 & X & Y & Z & 1 & -Xv & -Yv & -Zv \end{pmatrix}$ for my A matrix in $Ax = b$.

Setting b to a stacked matrix of all the x coordinates on top of the y coordinates, I used numpy.linalg.lstsq() to find an approximation for M. I then appended 1 to M and reshaped M to shape (3,4).

Part II: Estimating Object Points Robustly via RANSAC and the Fundamental Matrix

For part 2 I sampled 9 matches and called estimate_fundamental_matrix on them. The distance was found using $p1 * Fmatrix * p2.transpose()$ and every pair of matches that had a distance less than threshold 0.0005 was added as an inlier. Each iteration of the for loop also checked for the largest number of inliers and returned the best Fundamental matrix.

Part III: Converting 2D matches to 3D points

After getting stuck trying to create the coefficient matrix using numpy, I ended up using a for loop to go through each point and calculate the distance for each match, setting the A matrix in $Ax = b$ to

$$\begin{pmatrix} uM1_{31} - M1_{11} & uM1_{32} - M1_{12} & uM1_{33} - M1_{13} \\ vM1_{31} - M1_{21} & vM1_{32} - M1_{22} & vM1_{33} - M1_{23} \\ u'M2_{31} - M2_{11} & u'M2_{32} - M2_{12} & u'M2_{33} - M2_{13} \\ v'M2_{31} - M2_{21} & v'M2_{32} - M2_{22} & v'M2_{33} - M2_{23} \end{pmatrix}$$

and b to $\begin{pmatrix} M1_{14} - uM1_{34} \\ M1_{14} - vM1_{34} \\ M2_{14} - u'M2_{34} \\ M2_{14} - v'M2_{34} \end{pmatrix}$

I then used np.linalg.lstsq to find the 3D point and added it to the list of 3D points.

Part IV: Fundamental Matrix Estimation

For part 4 I constructed the coefficient matrix corresponding to
$\left(f_{11}uu' + f_{12}vu' + f_{13}u' + f_{21}uv' + f_{22}vv' + f_{23}v' + f_{31}u + f_{32}v + f_{33}\right) = 0$

Then I used linalg.svd to find Vh and set F to the last row of Vh. Then I reshaped F to shape (3,3) and used linalg.svd on F to find U, S, and Vh. I set the last element of S to 0, and then did $F = U * np.diagflat(S) * Vh$ to find the Fundamental matrix.

Originally, instead of setting F to the last row of Vh, I had set it to the row of Vh containing the smallest value in Vh. Similarly, for S I had originally set its smallest value to 0. However, the autograder in Gradescope said this implementation had too large of a distance, so I went with the other way. These lines of code are commented out.

Result

With a 0.0005 threshold of 3000 iterations I got:

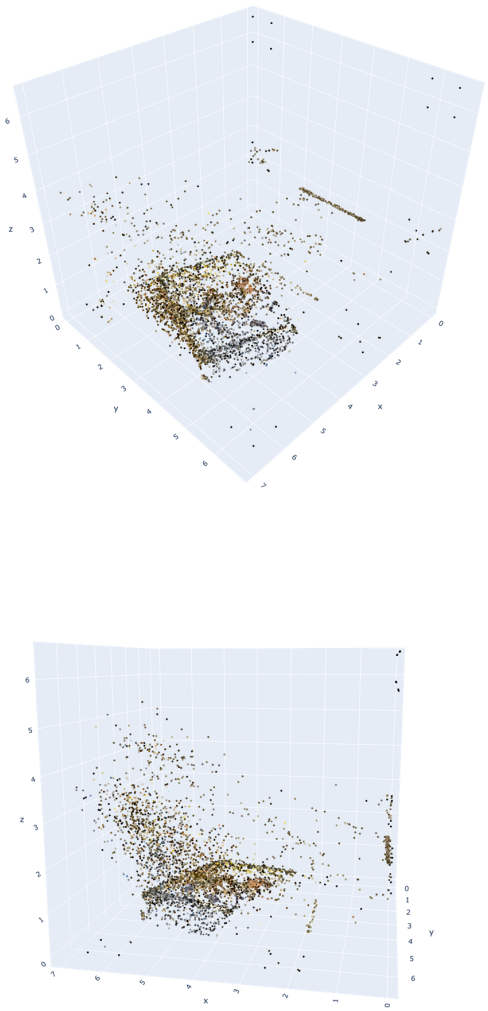

Figure 1: Cards

The cloud points contain the hard edges of the card box, but I noticed there were many points floating above the box. I played with the threshold, number of iterations, and number of keypoints but was not able to figure out how to get rid of them completely. Decreasing the threshold decreased the floating points but also decreased the number of accurate points on the card box. I increased the number of iterations to 3000 to get an output with more points. The foggy point cloud shows that my RANSAC implementation doesn't filter out all non-matches, but it was able to produce an identifiable point cloud

for the box of cards. The Mike and Ike boxes, on the other hand, were harder to make out in their point clouds here:

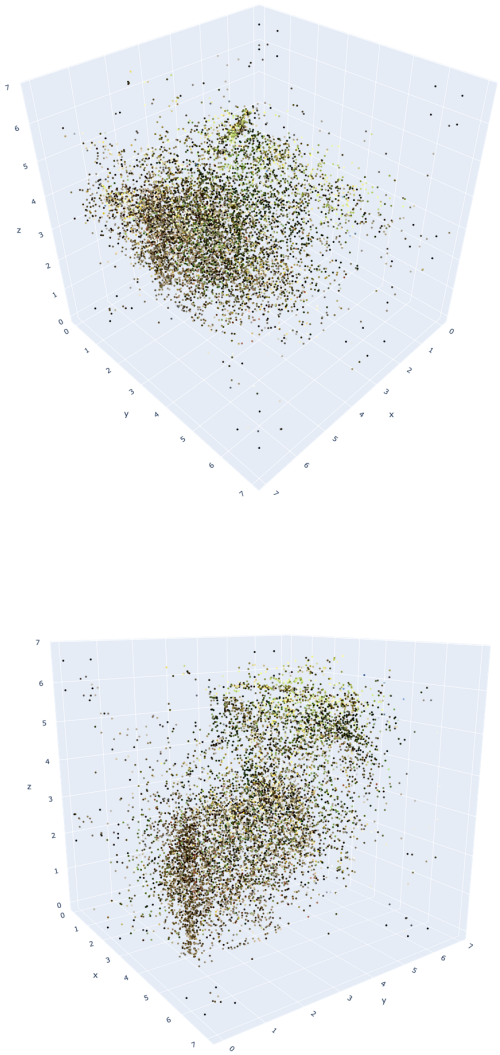

Figure 2: Mike and Ikes

After moving around the model I could see the box's edges but it was much more clouded than the cards. Looking at the Mike and Ike images, I think this point cloud is more ambiguous than the cards because the images are less similar to each other. I also noticed that the matches were less accurate since paired photos would be taken on opposite sides of the subject.