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

In this project, we learn how to estimate the camera projection matrix and the fundamental matrix. We use RANSAC to find the best fundamental matrix that gives the most inlier matches between two images. After matching, we reproject the 2D points back to 3D points to show points cloud of the object.

Implementation Detail

The implementation order follows the instructions on the project page.

Part I: Camera Projection Matrix

The implementation details can be checked in `calculate_projection_matrix`. The matrix A is constructed for every point pair of 2D and 3D points as the form shown in Eq. 1. Then M matrix is solved using numpy function `numpy.linalg.lstsq()` and reshaped to (3,4) after appending $m_{34} = 1$ in the end.

$$\begin{pmatrix} X & Y & Z & 1 & 0 & 0 & 0 & 0 & -Xu & -Yu & -Zu \\ 0 & 0 & 0 & 0 & X & Y & Z & 1 & -Xv & -Yv & -Zv \end{pmatrix} \begin{pmatrix} m_{11} \\ m_{12} \\ m_{13} \\ m_{14} \\ m_{21} \\ m_{22} \\ m_{23} \\ m_{24} \\ m_{31} \\ m_{32} \\ m_{33} \end{pmatrix} = \begin{pmatrix} u \\ v \end{pmatrix} \tag{1}$$

Part II: RANSAC

The implementation of RANSAC can be checked in `ransac_fundamental_matrix`. In general, the function iterates `num_iterations` to find the best estimated fundamental matrix. For a point correspondence (x', x), $x'Fx = 0$ if the fundamental matrix is perfect. Thus, in each iteration, inlier is counted if the error is smaller than the threshold, which is manually set as 1e-2 here.

Part III: Converting 2D matches to 3D points

Similar to Part I, we hardcode matrixes follow the format $Ax = b$ shown in Eq. 2 for each point correspondence.

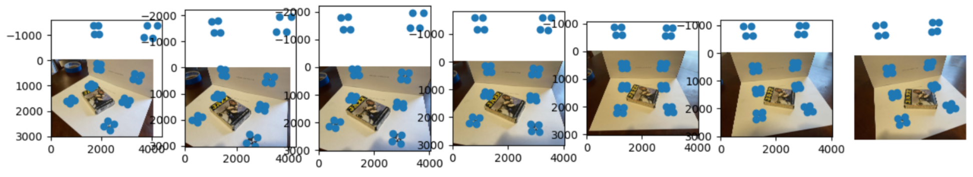
$$\begin{pmatrix} m_{11} - m_{31}u & m_{12} - m_{32}u & m_{13} - m_{33}u \\ m_{21} - m_{31}v & m_{22} - m_{32}v & m_{23} - m_{33}v \end{pmatrix} \begin{pmatrix} X \\ Y \\ Z \end{pmatrix} = \begin{pmatrix} m_{34}u - m_{14} \\ m_{34}v - m_{24} \end{pmatrix} \tag{2}$$

Part IV: Fundamental Matrix Estimation and Coordinate Normalization

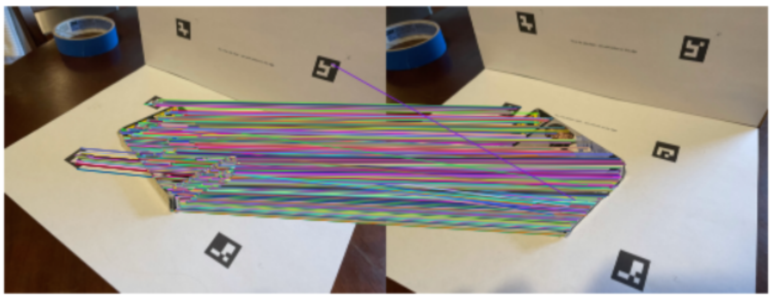
The implementation details can be checked in `estimate_fundamental_matrix`. In general, it follows the procedure in slides of 8-points methods. The normalization of coordinates has been implemented as well. However, the autograder said the function failed while it runs normal locally. For the normalization, the mean and standard deviation are calculated before the matrix transformation. The procedure follows instructions in Part V on project page.

Result

Result after implementing function `calculate_projection_matrix` of sequences cards.

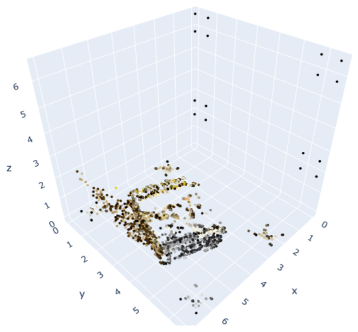


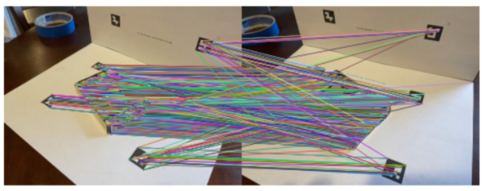
Result after implementing function `ransac_fundamental_matrix` for images 1 and 2 of sequences of cards.



Show us an image of a 3D reconstruction of one of the sequences (mike and ikes, cards, or dollars). Discuss your observations of the effect of any remaining incorrect matches.

Below Left is the 3D reconstruction of cards. We could find that there are many mismatches in left side. By checking all the intermediate figures showing matches in 2D, we found that the markers can lead to many mismatches. As markers are rigidly placed across the images in certain direction, it might can partially explained why so many mismatch in one direction.





Write about your thoughts on how well RANSAC does. How does the number of iterations affect the output?

From the figure below, we could see that the number of iterations will affect the final number of matches in 3D spaces. Fewer number of iterations might not be able to find the best estimated fundamental matrix and thus, cannot find so many matches.

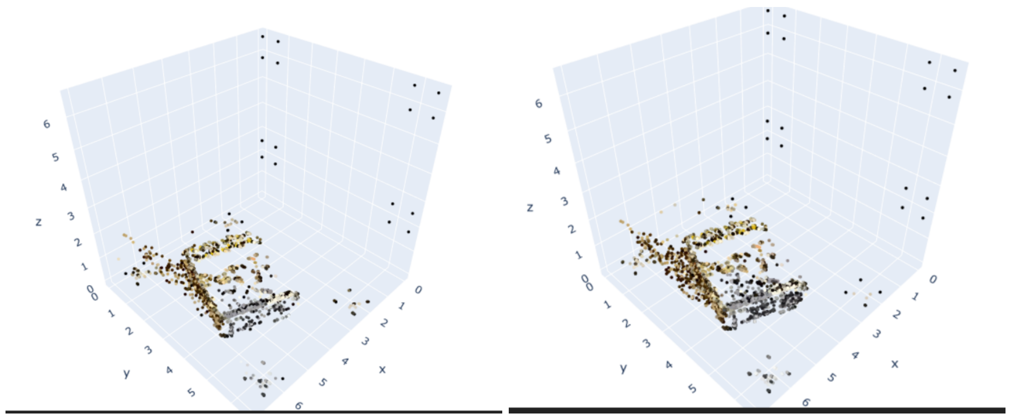


Figure 1: *Left*: the number of iteration in RANSAC is 1. *Right*: the number of iteration in RANSAC is 100.