SYDE 671 Advanced Image Processing

David Abou Chacra University of Waterloo Fall 2021

Assignment 5 - Coding Part
Camera Calibration and Fundamental Matrix Estimation with RANSAC

Instructions

- Your edits should be made in *"student.py"*
- Your submission should be (1) A jupyter notebook that includes both the report and your codes from student.py, (2) Your final student.py

Your report should include the following information:

- Describe your process and algorithm, show your results, describe any extra credit, and share any other information you feel is relevant.
- Report your estimate of the projection matrix and the camera center.
- Your estimate of the fundamental matrix for the base image pair (pic_a.jpg and pic_b.jpg).
- Several different images with the epipolar lines drawn on them and with the inlier keypoint correspondences shown. At least one of these pairs should be "correct" for full credit.

Overview

This project consists of three parts:

1. Estimating the projection matrix,
2. Estimating the fundamental matrix, and
3. Estimating the fundamental matrix reliably with RANSAC from unreliable SIFT matches.

These three tasks can be implemented in student.py.
The goal of this project is to introduce you to camera and scene geometry. Specifically we will estimate the camera projection matrix, which maps 3D world coordinates to image coordinates, as well as the fundamental matrix, which relates points in one scene to epipolar lines in another. The camera projection matrix and the fundamental matrix can each be estimated using point correspondences. To estimate the projection matrix—intrinsic and extrinsic camera calibration—the input is corresponding 3d and 2d points.

To estimate the fundamental matrix the input is corresponding 2d points across two images. You will start out by estimating the projection matrix and the fundamental matrix for a scene with

ground truth correspondences. Then you'll move on to estimating the fundamental matrix using point correspondences from SIFT and RANSAC.
Remember the challenging images of Gaudi's Episcopal Palace from Assignment 2?! By using RANSAC to find the fundamental matrix with the most inliers, we can filter away spurious matches and achieve near perfect point to point matching (hopefully!).

Part 1: Camera Projection Matrix

Given known 3D to 2D point correspondences, how can we recover a projection matrix that transforms from world 3D coordinates to 2D image coordinates? Recall that using homogeneous coordinates the equation for moving from 3D world to 2D camera coordinates is:

$$\begin{pmatrix} u \\ v \\ 1 \end{pmatrix} \cong \begin{pmatrix} su \\ sv \\ s \end{pmatrix} = M \begin{pmatrix} X \\ Y \\ X \\ 1 \end{pmatrix} = \begin{pmatrix} m_{1,1} & m_{1,2} & m_{1,3} & m_{1,4} \\ m_{2,1} & m_{2,2} & m_{2,3} & m_{2,4} \\ m_{3,1} & m_{3,2} & m_{3,3} & m_{3,4} \end{pmatrix} \begin{pmatrix} X \\ Y \\ X \\ 1 \end{pmatrix} \tag{1}$$

Your task is to find M by setting up a system of linear equations to find the least squares regression solution for these camera matrix parameters, given correspondences between 3D world points and 2D image points. The above equations can be rewritten so that you get two constraints per point correspondance. One from *u* and one from *v*:

$$u = \frac{m_{11}X + m_{12}Y + m_{13}Z + m_{14}}{m_{31}X + m_{32}Y + m_{33}Z + m_{34}}$$

$$\rightarrow (m_{31}X + m_{32}Y + m_{33}Z + m_{34})u = m_{11}X + m_{12}Y + m_{13}Z + m_{14}$$

$$\boxed{\rightarrow 0 = m_{11}X + m_{12}Y + m_{13}Z + m_{14} - m_{31}uX - m_{32}uY - m_{33}uZ - m_{34}u} \tag{eq1}$$

$$v = \frac{m_{21}X + m_{22}Y + m_{23}Z + m_{24}}{m_{31}X + m_{32}Y + m_{33}Z + m_{34}}$$

$$\rightarrow (m_{31}X + m_{32}Y + m_{33}Z + m_{34})v = m_{21}X + m_{22}Y + m_{23}Z + m_{24}$$

$$\boxed{\rightarrow 0 = m_{21}X + m_{22}Y + m_{23}Z + m_{24} - m_{31}vX - m_{32}vY - m_{33}vZ - m_{34}v} \tag{eq2}$$

With these rearrangements, we're able to set up our linear regression to find the elements of the matrix M. However, in projective geometry, the matrix M is only defined up to an unknown scale. In effect, these equations have many different possible solutions. In the degenerate solution case, M = all zeros, which is not very helpful in our context. We will consider two ways around this:

- We can fix the scale by setting the last element m34 to 1 and then find the remaining coefficients, or

- We can use the singular value decomposition to directly solve the constrained optimization problem in which the scale is set:

$$\begin{aligned} \min \quad & \|Ax\| \\ \text{s.t.} \quad & \|x\| = 1 \end{aligned} \tag{2}$$

Having recovered a vector of values with one of these methods, we must reshape it into the estimated camera projection matrix M. To help you validate M, we provide evaluation code which computes the total ”residual error” between the projected 2d location of each 3d point and the actual location of that point in the 2d image. The residual is the Euclidean distance in the image plane (square root of the sum of squared differences in u and v). This should be very small—in the order of a pixel. We also provide a set of “normalized points” in the files pts2d-norm-pic_a.txt and pts3d-norm.txt. The starter code will load these 20 corresponding normalized 2D and 3D points. If you solve for M using all the points, you should receive a matrix that is a scaled equivalent of the following matrix:

$$\begin{pmatrix} -0.4583 & 0.2947 & 0.0139 & -0.0040 \\ 0.0509 & 0.0546 & 0.5410 & 0.0524 \\ -0.1090 & -0.1784 & 0.0443 & -0.5968 \end{pmatrix} \tag{3}$$

Given this matrix, we can project 3D points in the world onto our camera plane. For example, this matrix will take the normalized 3D point $(1.2323, 1.4421, 0.4506, 1.0)$ and project it to 2D image (u, v) of $(0.1419, -0.4518)$ (after converting the homogeneous 2D point (us, vs, s) to its nonhomogeneous version by dividing by s).

Once we have an accurate projection matrix M, it is possible to take it apart into the more familiar and more useful matrix K of intrinsic parameters and matrix $[R\|T]$ of extrinsic parameters. For this assignment we will only ask you to estimate one particular extrinsic parameter: the camera center in world coordinates. Let us define M as being made up of a 3×3 matrix called Q, with 4th column m_4:

$$M = (Q|m_4), \tag{4}$$

The center of the camera C could be found by:

$$C = -Q^{-1}m_4 \tag{5}$$

If we use the normalized 3D points and M given above, we would receive camera center: C_{normA}=$(-1.5125, -2.3515, 0.2826)$.

You’re also provided a visualization which will show the estimated 3d location of the camera with respect to the normalized 3d point coordinates.

Part II: Fundamental Matrix Estimation

The next part of this project is to estimate the mapping of points in one image to lines in another by means of the fundamental matrix. This will require you to use similar methods to those in part 1. We will make use of the corresponding point locations listed in pts2d-pic_a.txt and pts2d-pic_b.txt. Recall that the definition of the fundamental matrix is:

$$\begin{pmatrix} u' & v' & 1 \end{pmatrix} \begin{pmatrix} f_{11} & f_{12} & f_{13} \\ f_{21} & f_{22} & f_{23} \\ f_{31} & f_{32} & f_{33} \end{pmatrix} \begin{pmatrix} u & v & 1 \end{pmatrix} = 0 \tag{6}$$

The fundamental matrix is sometimes defined as the transpose of the above matrix with the left and right image points swapped. Both are valid fundamental matrices, but the visualization functions in the starter code assume you use the above form.

Another way of writing this matrix equations is:

$$\begin{pmatrix} u' & v' & 1 \end{pmatrix} \begin{pmatrix} f_{11}u + f_{12}v + f_{13} \\ f_{21}u + f_{22}v + f_{23} \\ f_{31}u + f_{32}v + f_{33} \end{pmatrix} = 0 \tag{7}$$

Which is the same as:

$$(f_{11}uu' + f_{12}vu' + f_{13}u' + f_{21}uv' + f_{22}vv' + f_{23}v' + f_{31}u + f_{32}v + f_{33}) \tag{8}$$

With this, we can build a linear system to solve with least squares regression. For each unknown variable, we need one equation to constrain our solution. A pair of corresponding points will produce one equation for our system. However, as in part I, this matrix is only defined up to scale and the degenerate zero solution solves these equations. We need to fix the scale and solve using the same methods as in part 1. This leaves us able to solve the system with 8 point correspondences (or more, as we recover the best least squares fit to all points).
The least squares estimate of F is full rank; however, the fundamental matrix is a rank 2 matrix. As such we must reduce its rank. To do this, we can decompose F using singular value decomposition into the matrices UΣV'=F. We can then estimate a rank 2 matrix by setting the smallest singular value in Σ to zero thus generating Σ_2. The fundamental matrix is then easily calculated as F = U Σ_2 V'. We can check our fundamental matrix estimation by plotting the epipolar lines using the plotting function provided in the starter code.

Coordinate Normalization Notes Your estimate of the fundamental matrix can be improved by normalizing the coordinates before computing the fundamental matrix. It is suggested you perform the normalization through linear transformations as described below to make the mean of the points zero and the average magnitude about 1.0 or some other small number ($\sqrt{2}$ is also recommended). The normalizing transformation T is made from two parts, a scaling one and a translation one:

$$\begin{pmatrix} \hat{u} & \hat{v} & 1 \end{pmatrix} = \begin{pmatrix} s & 0 & 0 \\ 0 & s & 0 \\ 0 & 0 & 1 \end{pmatrix} \begin{pmatrix} 1 & 0 & -c_u \\ 0 & 1 & -c_v \\ 0 & 0 & 1 \end{pmatrix} \begin{pmatrix} u & v & 1 \end{pmatrix} \tag{9}$$

The transform matrix T is the product of the scale and offset matrices. c_u and c_v are the mean coordinates. To compute a scale s you could estimate the standard deviation after substracting the means. Then the scale factor s would be the reciprocal of whatever estimate of the scale you are using. Or you could find the maximum absolute value. Or you could scale the coordinates such that the average squared distance from the origin (after centering) is 2.

You could use one scale matrix based on the statistics of the coordinates from both images or you could do it per image. However you scale your coordinates, you will need to use the scaling matrices to adjust your fundamental matrix so that it can operate on the original pixel coordiantes as follows:

$$F_{orig} = T_b^T F_{norm} T_a. \tag{10}$$

This can be implemented in your function *estimate_fundamental_matrix()* written for part II, but you won't actually notice a difference in part II because the input correspondences are pretty much perfect. In part III (which calls estimate_fundamental_matrix()) you are more likely to notice an improvement. Alternatively, you could implement the scaling based on the distribution of all feature coordinates and not just the handful passed into *estimate_fundamental_matrix()*. In your report you should highlight one before and after case where normalization improved your results.

Part III: Fundamental Matrix with RANSAC

Given two photographs of a scene, it is unlikely that our point corresponence with which to perform regression for the fundamental matrix would all be correct. Your next task is to use RANSAC to reliably estimate a fundamental matrix from unreliable point correspondences computed with a feature point detector like SIFT.

The starter code uses the OpenCV library to perform feature point matching with the ORB descriptor for an image pair. We'll use these possible point correspondences and RANSAC to try and find a good fundamental matrix. We will iteratively choose a random set of point correspondences (e.g., 8, 9, or some small number of points), solve for the fundamental matrix using the function you wrote for part II, and then count the number of inliers. Inliers in this context will be point correspondences that ”agree” with the estimated fundamental matrix.

To count how many inliers a fundamental matrix has, you will need a distance metric based on the fundamental matrix. (Hint: For a point correspondence (x',x) what properties does the fundamental matrix have?).

You will also need to pick a threshold between inliers and outliers. The results will be very sensitive to this threshold, so explore a range of values which strikes a balance between permission and exclusion. The class notes has some hints on how to systematically pick this threshold.

You must also decide on a stopping criteria for RANSAC, e.g., considering thousands of iterations of RANSAC. Upon stopping, return the fundamental matrix with the most inliers.
Recall from lecture the expected number of iterations of RANSAC to find the ”right” solution in the presence of outliers. For example, if half of your input correspondences are wrong, then you have a 0.5^8 = 0.39% chance to randomly pick 8 incorrect correspondences when estimating the fundamental matrix. The correct fundamental matrix must have more inliers than outliers created from spurious matches, else the problem is not solvable.

Limitations Note For some real images, the fundamental matrix that is estimated may imply an impossible relationship between the cameras, e.g., an epipole in the center of one image when the cameras actually have a large translation parallel to the image plane. The estimated fundamental matrix may also be incorrect because the world points are coplanar (for planar scenes we need to use homographies - Fundamental matrices are degenerate), or because the cameras are not actually pinhole cameras and have lens distortions. Still, these ”incorrect” fundamental matrices tend to remove incorrect feature point matches (and, unfortunately, many correct ones too).

Evaluation and Visualization

For part I, we have provided expected output (matrix M and camera center C). These are numerical estimates so we won't be checking for exact numbers, just approximately correct locations.

For part II, you will be evaluated based on your estimate of the fundamental matrix. You can test how good your estimate of the fundamental matrix is by drawing the epipolar lines on one image which correspond to a point in the other image. You should see all of the epipolar lines crossing through the corresponding point in the other image, as shown in figure 1. For every point you pick in one image, you should be able to draw the epipolar line in the second image. The line should pass through your selected points' match in the new image.

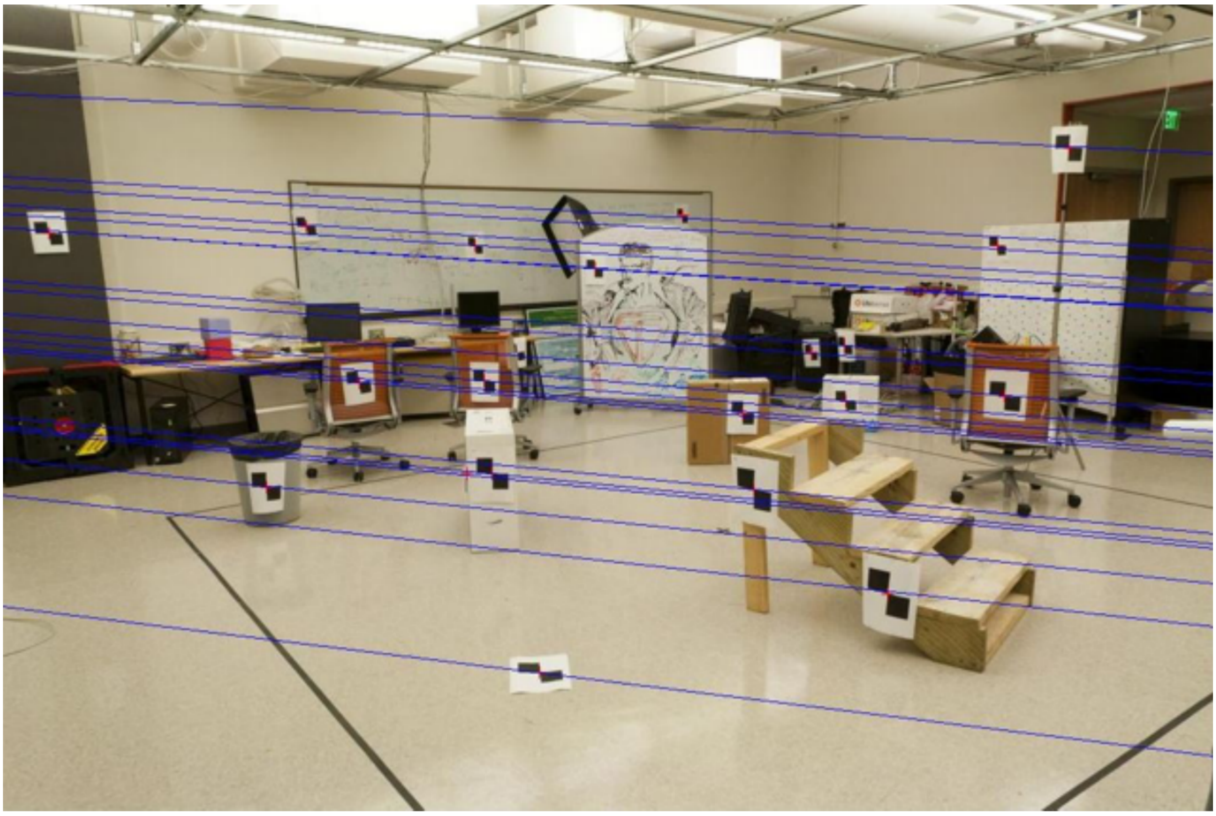

Figure 1: epipolar lines corresponding to points selected from another image.

You are provided with a function in the starter code that draws an epipolar line in an image given the fundamental matrix, and a point from the other image.

Part III is the most open ended part of this assignment. Your goal should be to demonstrate that you can estimate a fundamental matrix for a real image pair and use it to reject spurious keypoint matches. You should visualize the epipolar lines in your report. The Gaudi image pair shown above is fairly difficult and you might not be able to find a reasonable fundamental matrix without the coordinate normalization described above. Notre Dame is also difficult because the keypoint matches are mostly planar and thus the fundamental matrix is not well constrained.

We do not include the keypoint accuracy evaluation used in assignment 2. You should be able to get near-perfect accuracy (very few outliers) among the keypoints you designate as inliers.

Banned Functions

You must construct the matrices for camera calibration and F matrix estimation yourself, but you can use off-the-shelf linear solvers. You also must implement RANSAC yourself.

Rubric

- 12 pts: Correctly setting up the system of equations for the least squares regression for the projection matrix.
- 12 pts: Correctly solving for the projection matrix and correctly solving for the camera center.
- 12 pts: Correctly setting up the fundamental matrix regression.
- 12 pts: Correctly solving for the fundamental matrix and generating good epipolar lines on the test set.
- 27 pts: Correctly combining RANSAC with fundamental matrix estimation and generating epipolar lines on the test images.
- 25 pts: For a detailed and clear Report

Credit

This project is based on a project from Aaron Bobick's offering of CS 4495 at GATech, and has been expanded and edited by Henry Hu, Grady Williams, and James Hays. Eleanor Tursman adapted the code for Python, with help from Anna Sabel.