Project 4: Camera Calibration and Fundamental Matrix Estimation with RANSAC

CS 342 (Computer Vision)
Spring 2020
RKMVERI
Due: 11:59 pm on Monday, May 04, 2020

Overview

The goal of this project is to introduce you to camera and scene geometry. Specifically we will estimate the camera projection matrix or calibration 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 (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 ORB, which is an alternative to SIFT.

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 as shown below:

Setup

- Download project zip file from Google Classroom and unzip
- Run the notebook using:
  `jupyter notebook ./code/proj4.ipynb`
- Implement the required methods (described below) in code/student_code.py file
- Generate the submission once you've finished the project using:
  `python zip_submission.py`

Details and Starter Code

For this project, you need to implement three major parts:

- Estimating the projection matrix: `calculate_projection_matrix()`, `calculate_camera_center()`

- Estimating the fundamental matrix: `estimate_fundamental matrix()`
- Estimating the fundamental matrix with unreliable ORB matches using RANSAC: `ransac_fundamental_matrix()` (see Szeliski 6.1.4)

Part I: Camera Projection Matrix

The goal is to compute the projection matrix that goes 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:

$$\begin{pmatrix} u \\ v \\ 1 \end{pmatrix} \cong \begin{pmatrix} u*s \\ v*s \\ s \end{pmatrix} = \begin{pmatrix} m_{11} & m_{12} & m_{13} & m_{14} \\ m_{21} & m_{22} & m_{23} & m_{24} \\ m_{31} & m_{32} & m_{33} & m_{34} \end{pmatrix} \begin{pmatrix} X \\ Y \\ Z \\ 1 \end{pmatrix}$$

Another way of writing this equation is:

$$\begin{aligned} u &= \frac{m_{11}X + m_{12}Y + m_{13}Z + m_{14}}{m_{31}X + m_{32}Y + m_{33}Z + m_{34}} \\ &\rightarrow (m_{31}X + m_{32}Y + m_{33}Z + m_{34})u = m_{11}X + m_{12}Y + m_{13}Z + m_{14} \\ &\rightarrow 0 = m_{11}X + m_{12}Y + m_{13}Z + m_{14} - m_{31}uX - m_{32}uY - m_{33}uZ - m_{34}u \end{aligned}$$

$$\begin{aligned} v &= \frac{m_{21}X + m_{22}Y + m_{23}Z + m_{24}}{m_{31}X + m_{32}Y + m_{33}Z + m_{34}} \\ &\rightarrow (m_{31}X + m_{32}Y + m_{33}Z + m_{34})v = m_{21}X + m_{22}Y + m_{23}Z + m_{24} \\ &\rightarrow 0 = m_{21}X + m_{22}Y + m_{23}Z + m_{24} - m_{31}vX - m_{32}vY - m_{33}vZ - m_{34}v \end{aligned}$$

At this point, you're almost able to set up your linear regression to find the elements of the matrix M. There's only one problem, the matrix M is only defined up to a scale. So these equations have many different possible solutions, in particular M = all zeros is a solution which is not very helpful in our context. The way around this is to first fix a scale and then do the regression. There are several options for doing this (1) You can fix the last element

m_{34} to 1 and then find the remaining coefficients, or you can use the singular value decomposition to directly solve the constrained optimization problem:

$$\min \|Ax\| \quad s.t \, \|x\| = 1$$

To make sure that your code is correct, we are going to give you a set of "normalized points" in the files `./pts2d-norm-pic_a.txt` and `/pts3d-norm.txt`. If you solve for M using all the point you should get a matrix that is a scaled equivalent of the following (the negative of this matrix is a scaled equivalent):

$$M_{\text{norm}A} = \begin{pmatrix} -0.4583 & 0.2947 & 0.0139 & -0.0040 \\ 0.0509 & 0.0546 & 0.5410 & 0.0524 \\ -0.1090 & -0.1784 & 0.0443 & -0.5968 \end{pmatrix} \quad (1)$$

For example, this matrix will take the last normalized 3D point which is $< 1.2323, 1.4421, 0.4506, 1.0 >$ and will project it to the $< u, v >$ of $< 0.1419, -0.4518 >$ where we converted the homogeneous 2D point $< us, vs, s >$ to its inhomogeneous version (the transformed pixel coordinate in the image) by dividing by s

The first task for you is to write the least squares regression to solve for M given the corresponding normalized points. The starter code will load 20 corresponding normalized 2D and 3D points. You have to write the code to set up the linear system of equations, solve for the unknown entries of M, and reshape it into the estimated projection matrix. To validate that you've found a reasonable projection matrix, we've provided evaluation code which computes the total "residual" between the projected 2d location of each 3d point and the actual location

of that point in the 2d image. The residual is just the distance (square root of the sum of squared differences in u and v). This should be very small.

Once you have an accurate projection matrix M, it is possible to tease it apart into the more familiar and more useful matrix K of intrinsic parameters and matrix $[R|T]$ of extrinsic parameters. For this project 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 3x3 we'll call Q and a column will call m_4:

$$M = (Q|m_4)$$

From class we said that the center of the camera C could be found by:

$$C = -Q^{-1}m_4$$

To debug your code: If you use you the normalized 3D points to get the M given above you would get a camera center of:

$$C_{\text{norm}A} =< -1.5125, -2.3515, 0.2826 >$$

We've 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 estimating 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:

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

Note: 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.

And another way of writing this matrix equations is:

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

Which is the same as:

$$\left(f_{11}uu' + f_{12}vu' + f_{13}u' + f_{21}uv' + f_{22}vv' + f_{23}v' + f_{31}u + f_{32}v + f_{33}\right) = 0$$

Starting to see the regression equations? Given corresponding points you get one equation per point pair. With 8 or more points you can solve this (why 8?). As in part I there's an issue here where the matrix is only defined up to scale and the degenerate zero solution solves these equations. So you need to solve using the same method you used in part 1 of first fixing the scale and then solving the regression.

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. In order to do this we can decompose F using singular value decomposition into the matrices $UDV' = F$. We can then estimate a rank 2 matrix by setting the smallest singular value in D to zero thus generating D_2 . The fundamental matrix is then easily calculated as $F = UD_2V'$. You can check your fundamental matrix estimation by plotting the epipolar lines using the plotting function provided in the starter code.

Part III: Fundamental Matrix with RANSAC

For two photographs of a scene it's unlikely that you'd have perfect point corresponence with which to do the regression for the fundamental matrix. So, next you are going to compute the fundamental matrix with unreliable point correspondences computed with ORB. As discussed in class, least squares regression alone is not appropriate in this scenario due to the presence of multiple outliers. In order to estimate the fundamental matrix from this noisy data you'll need to use RANSAC in conjunction with your fundamental matrix estimation.

You'll use these putative point correspondences and RANSAC to find the "best" fundamental matrix. You will iteratively choose some number of point correspondences (8, 9, or some small number), 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. In order to count how many inliers a fundamental matrix has, you'll need a distance metric based on the fundamental matrix. (Hint: For a point correspondence (x',x) what properties does the fundamental matrix have?). You'll need to pick a threshold between inlier and outlier and your results are very sensitive to this threshold so explore a range of values. You don't want to be too permissive about what you consider an inlier nor do you want to be too conservative. Return the fundamental matrix with the most inliears.

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 correspondences when estimating the fundamental matrix. Hopefully that correct fundamental matrix will have more inliers than one created from spurious matches, but to even find it you should probably do thousands of iterations of RANSAC.

Evaluation and Visualization

For part I we've told you the 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'll 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, like this:

We provide you 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. We may check the fundamental matrix you estimate and you should have a visualization of the epipolar lines in your writeup.

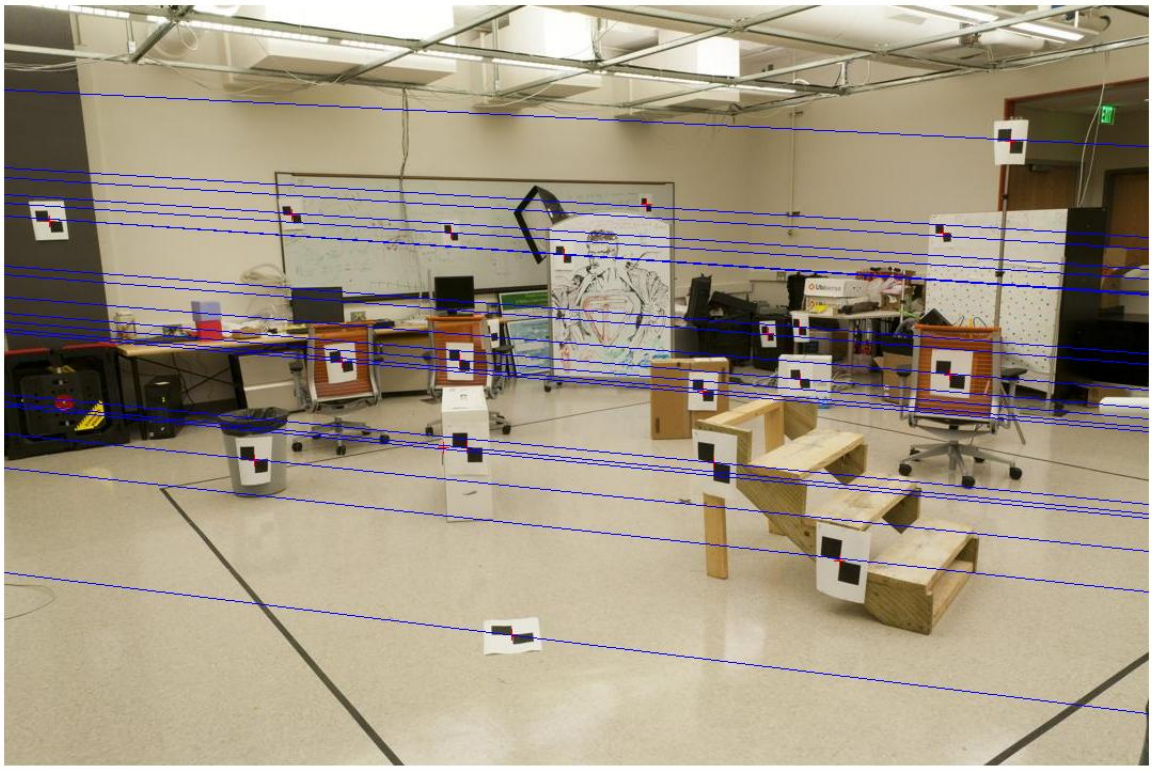
You should be able to get nearly perfect accuracy (very few outliers) among the keypoints you designate as inliers.

Data

You'll be provided with 2D and 3D ground truth point correspondences for the base image pair (pic_a.jpg and pic_b.jpg), as well as other images which will not have any ground truth dataset.

Useful Functions

- np.linalg.svd(): This function returns the singular value decomposition of a matrix. Useful for solving the linear systems of equations you build and reducing the rank of the fundamental matrix.
- np.linalg.inv(): This function returns the inverse of a matrix.
- np.random.randint(): Let's you pick integers from a range. Useful for RANSAC.



Banned Functions

You may also not using anyone else's code that estimates the fundamental matrix or performs RANSAC for you.

Write up

For this project, and all other projects, you must do a project report in PDF. In the report you will describe your algorithm and any decisions you made to write your algorithm a particular way. Discuss any extra faetures you implemented and show what contribution it had on the results

Make sure to include the following in your write-up:

1. Your estimate of the projection matrix and the camera center.
2. Your estimate of the fundamental matrix for the base image pair (pic_a.jpg and pic_b.jpg)
3. Several different images with the epipolar lines drawn on them and with the inlier keypoint correspondences shown.

Handing in

This is very important as you will lose points if you do not follow instructions. Every time you do not follow instructions, you will lose 5 points. The folder you hand in must contain the following:

- README.txt - text file containing how to run your code, along with anything about the project that you want to tell the instructor
- code/ - directory containing all your code for this assignment
- results/ - directory containing the resultant images and PDF report

Turn in your project through Google Classroom.

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.
- +40 pts: Correctly combining RANSAC with fundamental matrix estimation and generating epipolar lines on the test images.
- +12 pts: Writeup
- +10 pts: Extra Credit (If you do something real fancy)
- -5*n pts: Lose 5 points for every time you do not follow the instructions for the hand in format

Credits

This project is taken from a similar project from Aaron Bobick's offering of 4495 and James Hays's offering of CS 6476 at Georgia Tech.