CMSC 426, Computer Vision
Project 4: Camera Calibration and Epipolar Geometry

Prof. Yiannis Aloimonos,
Jack Rasiel and Kaan Elgin

April 10, 2018

1 Due Dates

This project is not divided into milestones.

- **DUE DATE:** 11:59:59PM on Tuesday, **March 24** 2018

2 Introduction

This project focuses on on some foundational concepts and techniques for relating points in an image with points in the world. Specifically, you'll estimate the camera projection matrix (mapping points in the world to points in the image), and the fundamental matrix (mapping between points and lines in two images of the same scene).

The aim of this project is to build a rigorous understanding of the concepts involved, as well as introduce some practical techniques for using them.

This project has two parts:

- Camera Calibration from 2D-3D point correspondences in images.
- Fundamental Matrix Estimation (with RANSAC for refinement)

More details are given below. Note that these concepts were introduced in class, and not all details are repeated here. Please see the course slides for reference, or some of the other resources linked in the "Helpful Resources" section.

3 Camera Calibration: Estimating the Projection Matrix (34 pts)

The projection matrix maps the 3D coordinates in the world to 2D coordinates in the image. We can estimate it given a set of points in an image, and their corresponding 3D points in the world. Using homogeneous coordinates, we map from a point (X, Y, Z) in world coordinates, to (u, v, 1) in image coordinates, via the projection matrix M:

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

Your first task is to estimate the elements of M. This can be accomplished by solving a constrained optimization via least squares. The problem can be stated as follows:

$$\min \|Ax\|$$

$$s.t \ \|x\| = 1$$

(Note, we'll discuss the x = 1 constraint later.) Where m are the elements of M, and A is the set of observations (point correspondences). Each point correspondence is represented by two rows in A, one for u and one for v. We can find the format of these rows by manipulating formula 1 as follows:

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

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

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

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

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

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

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

Because the projection matrix M is only defined up to scale, the above equations can have many solutions– including just setting all the elements of M to zero, which wouldn't be very helpful! To avoid this, you can set the last element m34 to 1 and solve for the remaining coefficients, or leave m34 free but impose the constraint $||x|| = 1$ when solving.

3.1 What you need to do:

We've provided starter code to load a set of point correspondences. You need to set up the system of equations, solve for M, and reshape the results into the proper 3x4 matrix format. To test your code, we've included a set of corresponding 2D and 3D points. Given those points, your code should produce a matrix that is a scaled equivalent of the following:

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

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.

4 Estimating the Fundamental Matrix (33pts)

Given two different images of the same scene, the fundamental matrix maps points in one image to lines in the other. This is the relationship illustrated by the familiar epipolar diagram:

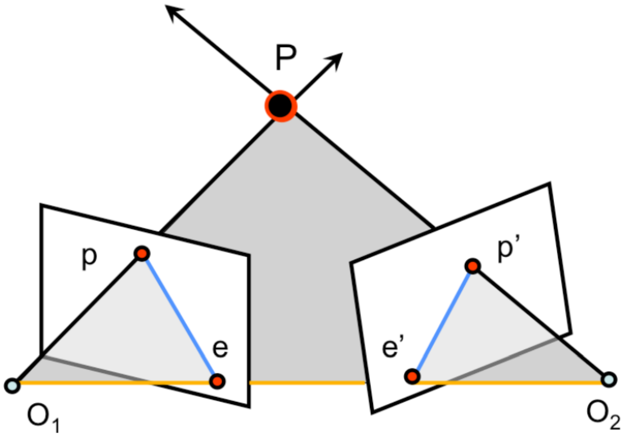

Figure 1: Two images of the same point P. The point p in camera 1 maps to the line $\overline{e'p'}$ in camera 2, and p' to $\overline{ep}$. *Source*.

We're going to estimate the fundamental matrix for a pair of images, using a set of corresponding points between them. To start, let's recall the definition of the fundamental matrix: [1]

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

This can simplified as follows:

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

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

As when we estimated the projection matrix in part 1, we can use multiple such equations to set up a linear system and solve for the elements of the matrix F. How many equations do we need? If we want to solve directly for all nine elements of F, we need nine equations to constrain them. However, as with the projection matrix, the fundamental matrix is only defined up to scale, and we need to avoid the unhelpful solution of setting that scalar to zero (and thus making all the elements of F zero). So, we fix the scale as in part 1. Thus we need at least eight equations (meaning, eight point correspondences).

If estimating F via least squares, the solution found may have full rank. However, we know that the fundamental matrix has rank 2. We should reduce the rank of our estimate to 2. To do this, we can decompose F using singular value decomposition into the matrices $U\Sigma V' = F$. 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 $F = U\Sigma_2 V'$.

We can check our fundamental matrix estimation by plotting the epipolar lines using the plotting function provided in the starter code.

4.1 What you need to do:

Fill in the template `estimate_fundamental_matrix.m`, adding code to estimate the fundamental matrix given point correspondences.

---

[1] Our code assumes you use the formula given here. However it is also valid to defined the fundamental matrix as the transpose of this matrix, with the points (u,v,1), (u,'v',1) swapped.

5 Refining the Fundamental Matrix Estimate with RANSAC (33 pts)

In project 2, we matched feature points between images using a sum-squared-difference metric. This produced a lot of incorrect matches, so we used RANSAC to separate good matches from bad. We'll take a similar approach to find a set of good matches for computing the fundamental matrix.

5.1 What you need to do:

Find and match feature points in a given image pair (for example, using `detectSurfFeatures` and `matchFeatures`). Fill in the template for ransac_fundamental_matrix with your RANSAC code. This should be similar to the RANSAC you implemented for project 2. Instead of estimating a homography, we're estimating a 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?).

To visualize your results, we have provided a function draw_epipolar_lines which draws an epipolar line in an image, given the fundamental matrix and a point from the other image. 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:

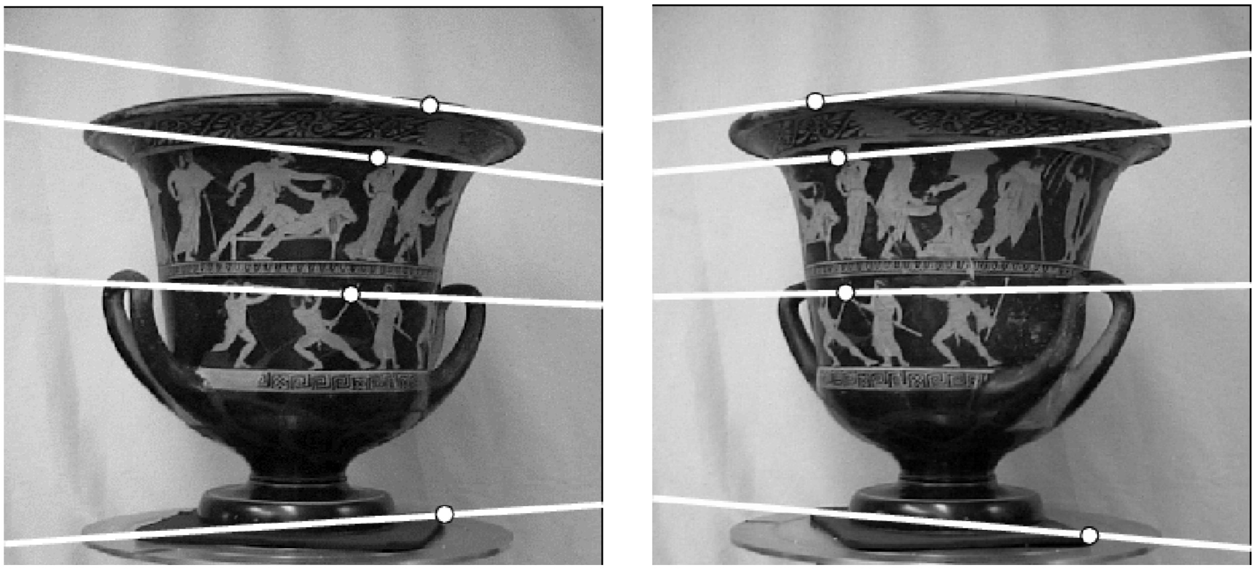

Figure 2: A very accurate fundamental matrix maps each keypoint in one image to a line in the other, which perfectly intersects the corresponding keypoint in that image. *Source: lecture slides*

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, or because the cameras are not actually pinhole cameras and have lens distortions. Still, these ”incorrect” fundamental matrices tend to remove incorrect feature matches (and, unfortunately, many correct ones too).

6 Extra Credit: Camera Calibration with a Chessboard (20pts)

What if we want to calibrate our camera, but don’t already have 2D-3D point correspondences? One approach is to calibrate using an object of known size and pattern, whose 3D position we can determine from its appearance in the image (such an object is known as a ”fiducial marker”). The most common calibration pattern is a rectangular ”chessboard”:

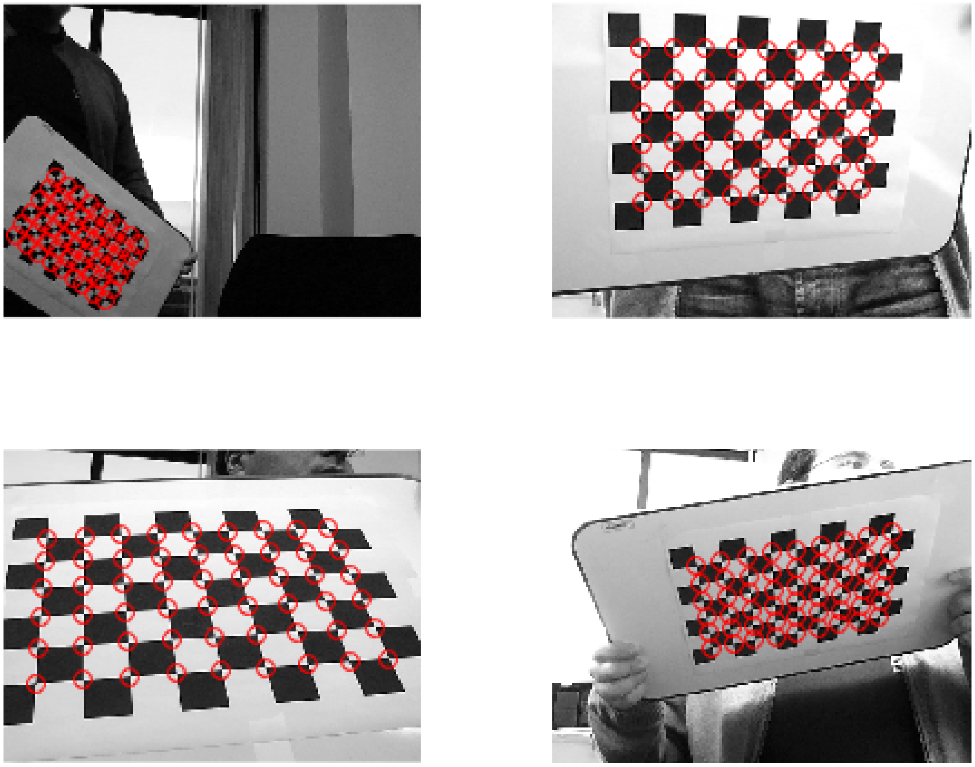

Figure 3: *source*

It’s easy to locate the chessboard’s clearly-defined corners in the image. Since the chessboard’s pattern is also known, we can determine its position and orientation relative to the camera. (Note that this position is only known up to a scale factor, unless we give the length (in mm/inches/etc) of one of the chessboard's squares.) We can then use these 2D-3D correspondences to calibrate our camera.

Since this is a common procedure for computer vision researchers, we'd like you to do it yourself, with your own camera and chessboard! You are encouraged to use the camera calibrator app from MATLAB's CV toolbox– it provides functionality for the tasks described below.

6.1 What you need to do:

Print out a chessboard of your own and collect some images. Use those images to estimate your intrinsics and extrinsics. In your report, you should include:

- a few of the images you used for calibration.
- Your estimated intrinsics and extrinsics.
- A visualization of the camera extrinsics.
- A visualization of the reprojection error for a few checkerboard images.
- The reprojection error bar graph.
- Answer: based on the reprojection error, do you think your calibration needs improvement?
- If there are outliers among your calibration images, remove them, and show how it affected the reprojection error (both in a few images and in the bar graph).

7 Submission Guidelines

If your submission does not comply with the following guidelines, you'll be given ZERO credit:

7.1 File tree and naming

Your submission on Canvas must be a zip file, following the naming convention **"YourDirectoryID_proj4.zip"**. For example, jrasiel_proj4.zip. The file **must have the following directory structure**, based on the starter files:

```
YourDirectoryID_proj4.zip
 |__code/
 |   |__proj3_part1.m
 |   |__proj3_part2.m
 |   |__proj3_part3.m
 |   |__calculate_projection_matrix.m
 |   |__draw_epipolar_lines.m
 |   |__estimate_fundamental_matrix.m
 |   |__ransac_fundamental_matrix.m
 |   |__(utility scripts that you don't need to modify:)
 |   |__evaluate_points.m
 |   |__plot3dview.m
 |   |__show_correspondence2.m
 |   |__show_correspondence.m
 |   |__visualize_points.m
 |__report.pdf - Your report.
```

7.2 Running your code

We expect your code to function as per the templates provided. Please do not modify the definitions of these functions. If you write code in external .m files, please include them in the `code/` directory.

7.3 Report:

For each section of the project, explain briefly what you did, and describe any interesting problems you encountered and/or solutions you implemented. You must include the following details in your writeup:

- Your estimate of the projection matrix.
- 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.

As usual, your report must be full English sentences, **not** commented code.

A reminder: Every student should present AT LEAST once in the class (you can volunteer to present for more than one project) and can choose to do for any of the projects.

8 Forbidden MATLAB Functions:

You may not use the constrained least squares version of the MATLAB lsqlin() function. You may also not using any tools from the MATLAB computer vision toolbox for RANSAC or for estimating the fundamental matrix.

9 Useful MATLAB Functions:

- `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.
- `inv()`. This function returns the inverse of a matrix.
- `randsample()`. Lets you pick integers from a range. Useful for RANSAC.
- For the initial feature detection and matching before RANSAC, we recommend `detectSURFFeatures` and `matchFeatures`.

10 Helpful Resources:

- Stanford course notes on epipolar geometry.
- Oxford course notes on epipolar geometry.
- And of course, the course lecture slides!

11 Acknowledgements

This project is based on a project from James Tompkin's CS143 at Brown, which itself was based on a project from Aaron Bobick's offering of CS 4495 at GATech.

12 Collaboration Policy

You are restricted to discuss the ideas with at most two other people. But the code you turn-in should be your own, and should be the result of you exercising your own understanding of it. If you reference anyone else's code in writing your project, you must properly cite it in your code (in comments) and your writeup. For the full honor code refer to the CMSC426 Spring 2018 website.

DON'T FORGET TO HAVE FUN AND PLAY AROUND WITH IMAGES!