16720 Computer Vision: Homework 5
3D Reconstruction

Instructor: Martial Hebert
TAs: Xinlei Chen, Arne Suppé, Jacob Walker, Feng Zhou
Due Date: November 13th, 11:59:59.$\bar{9}$ p.m.

Instructions/Hints

1. Please pack your system and write-up into a single file named **<AndrewId>.zip**.
2. All questions marked with a **Q** require a submission.
3. For the implementation parts, please stick to the headers, variable names, and file conventions provided.
4. There are some difficult theory questions at the end. **Start early!** As always!
5. Attempt to verify your implementation as you proceed: If you don't verify that your implementation is correct on toy examples, you will risk having a huge mess when you put everything together.
6. If you have any question, please contact: **Jacob Walker**, jcwalker@cs.cmu.edu OR **Xinlei Chen**, enderchen@cs.cmu.edu.

1 Fundamental matrix estimation

Now you will explore different methods of estimating the fundamental matrix given a pair of images. Then, you will compute the camera matrices and triangulate the 2D points to obtain the 3D Scene structure. In the `temple/` directory, you will find two images of a temple.

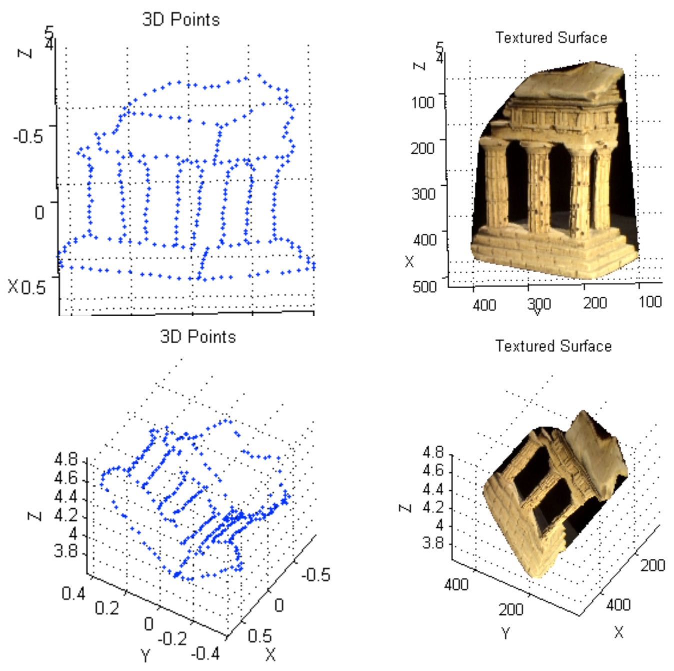

Figure 1: Two novel views of an object reconstructed in 3D from two images.

The Eight Point Algorithm

The 8-point algorithm discussed in class and outlined in *Section 10.1 of Forsyth & Ponce* is arguably the simplest method for estimating the fundamental matrix. For this section, you are supposed to manually select point correspondences using `cpselect` or use provided correspondances from `temple/some_corresp.mat`. Pass them to your implementation of the 8-point algorithm.

Q1.1 (10 points) Submit a function with the following signature for this portion of the assignment:

```
function F = eightpoint(X,Y,M);
```

where X and Y are each $N \times 2$ matrices with coordinates that constitute correspondences with the first and second image respectively. (the format returned by `cpselect`). M is the scaling factor. You should test your code on hand-selected correspondences for each image pair. Remember that the x-coordinate of a point in the image is its column entry, and y-coordinate is the row entry. Note: 8-point is just a figurative name. Your algorithm should use an over-determined system.

To visualize the correctness of your estimated F, use the function displayEpipolarF.

You should scale the data as was discussed in class, by dividing each coordinate by M, the largest image dimension. After computing F, you will have to "unscale" the fundamental matrix. Save F, M, and your points as `q1_1.mat`

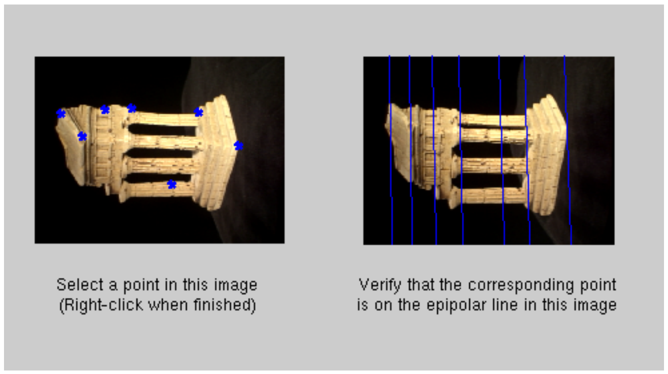
Select a point in this image
(Right-click when finished)
Verify that the corresponding point
is on the epipolar line in this image

Figure 2: Points on the epipolar lines.

The 7 Point Algorithm

Q1.2 (20 points) Since the fundamental matrix only has seven degrees of freedom, it is possible to calculate F using only seven point correspondences; this requires solving a polynomial equation.

In the section, you will implement the Seven Point Algorithm described in class outlined in Section 15.6 of Forsyth and Ponce, and in p. 350 in Szeliski. Use the point correspondences in the `7pt_corresp.mat` file to recover a fundamental matrix F. The function should have the signature:

```
function F = sevenpoint_norm(X, Y, M)
```

where `X` and `Y` are 7-by-2 matrices containing the correspondences, `M` is the normalizer, and F is a cell array of length either 1 or 3 containing Fundamental matrix/matrices. Use `M` to normalize the point values between $[0, 1]$ and remember to "unnormalize" your computed F afterwards.

Use `displayEpipolarF` to visualize F and pick the correct one. **In your answer sheet: write your recovered F and print an output of `displayEpipolarF`.**

Hints: if $x_{normalized} = Tx$, then $F_{unnormalized} = T^T F T$; you can use `roots`; the epipolar lines may not match exactly due to imperfectly selected correspondences. Save F and M as `q1_2.mat`

2 3D Reconstruction

Metric Reconstruction

The fundamental matrix can be used to determine the camera matrices of the two views used to generate F. In general, M_1 and M_2 will be projective camera matrices, i.e., they are related to each other by a projective transform. For details on recovering M_2, see 7.2 in Szeliski. Thankfully, we have provided you with the function `camera2(F, K1, K2, pts1, pts2)` that will find M_2 given F , K_1, and K_2 and a number of correspondence points. Use the correspondence points provided in `some_corresp.mat`. Use the intrinsic matrices K_i in `temple/intrinsics.mat`

Using the above, we have provided a function to triangulate a set of 2-D coordinates in the image to a set of 3-D points with the signature:

```
function P = triangulate(M1, pts_1, M2, pts_2)
```

where `pts_1` and `pts_2` are the 2-D image coordinates and `P` is a matrix with the corresponding 3-D points, per row. Make sure to multiply your M1, M2 by the camera matrices!

Q2.1 (10 points) Included in `hw5.zip`, is a file `many_corresp.mat` which contains 288 hand-selected points from `im1` saved in the variables `x1` and `y1`.

Now, we can determine the 3D location of these point correspondences using the triangulate function. These 3D point locations can then plotted using the MATLAB function scatter3. The resulting figure can be rotated using the Rotate 3D tool, which can be accessed through the figure menubar.

Please take a few screenshots of the 3D visualization, and include them with your homework submission.

3 3D Correspondence

To culminate this project, you will now calculate correspondences using 3D information via epipolar geometry.

Q3.1 (15 points) Write the function:

```
function [x2, y2] = epipolarCorrespondence(im1, im2, F, x1, y1)
```

This function takes in the x and y coordinates of a pixel on `im1`, and your fundamental matrix **F**, and returns the coordinates of the pixel on `im2` which correspond to the input point. Fortunately, we know the fundamental matrix **F**. Therefore, instead of searching for our matching point at every possible location in `im2`, we can simply search over the set of pixels that lie along the epipolar line (recall that the epipolar line passes through a single point in `im2` which corresponds to the point (`x1`, `y1`) in `im1`). Slide a window along the epipolar line and find the one that matches most closely to the window around the point in `im1`. There are various ways to compute the window similarity. For this assignment, simple methods such as the Euclidean or Manhattan distances between the intensity of pixels should suffice. Implementation hints:

- Experiment with various window sizes.
- It may help to use a Gaussian weighting of the window, so that the center has greater influence than the periphery.
- Since the two images only differ by a small amount, it might be beneficial to consider matches for which the distance from (x1, y1) to (x2, y2) is small.

To test your function epipolarCorrespondence, we have included a GUI:

```
function [coordsIM1, coordsIM2] = epipolarMatchGUI(im1, im2, F)
```

This script allows you to click on a point in im1, and will use your function to display the corresponding point in im2. The process repeats until you right-click in the figure, and the sets of matches will be returned. Use your function `epipolarCorrespondence` to calculate the corresponding points in `im2` from chosen points in `im1`. Save your points and F to `q3_1.mat`

4 Theory Questions : Epipolar Geometry (Xinlei)

Q4.1 (5 points) Consider the case of two cameras viewing an object such that the second camera differs from the first by a pure translation parallel to the image plane. Show that the epipolar lines in the two cameras are parallel.

Q4.2 (10 points) Show that the image of an object and the image from a camera with the same parameters of the same object viewed in a mirror are related by a skew-symmetric fundamental matrix, i.e., $\mathbf{F} = -\mathbf{F}^T$.

Q4.3 (5 points) Suppose two cameras fixate on a point P (see figure 3) in space such that their principal axes intersect at that point. Show that if the image coordinates are normalized so that the coordinate origin $(0, 0)$ coincides with the principal point, the $\mathbf{F}_{33}$ element of the fundamental matrix is zero.

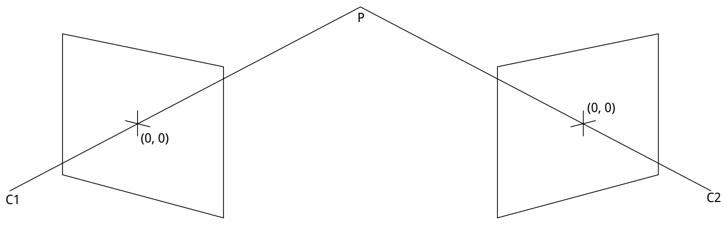

Figure 3: Figure for Q1.1. $C1$ and $C2$ are the optical centers. The principal axes intersect at point P.

Q4.4 (20 points) Homework 1 discussed two cases where a one-to-one mapping (homography) exists between a pair of images: a) when all the points of interest lie on a plane; b) when two cameras are separated by a pure rotation.

1. In the planar case, is it possible to find an epipolar line in the second image that corresponds to a point in the first image? If so, can you state the geometrical relation between it (the epipolar line) and the point found by homography? If not, give an explanation.

2. In the rotational case, answer the same question.