16-720J: Homework 5
RANSAC and 3D Reconstruction

Instructor - Gary Overett, TA - Yang Gao

13 November 2015 - Due Midnight Monday 30 November 2015

Instructions/Hints

1. Start early: This assignment involves plenty of implementation and cannot be debugged as easily since there are multiple inter-connected components.

2. **Submission:** Your submission should be a **single zip** file `NetID.zip` that at least contains:

   • A Root Level Folder: `<NetID>` - so that we can unpack your work and uniquely identify it!

   • `<NetID>.pdf`: a pdf containing your answers to all written items, including images or diagrams validating your technique. Your pdf should contain answers for Q1.*. It should also contain 4 snapshots of `displayEpipolarF` and 4 fundamental matrices you computed for Q2.1, Q2.2 and Q2.3 ($F_{8,clean}$, $F_{8,noisy}$, $F_{7,clean}$, $F_{7,RANSAC}$ respectively). Finally, a clear explanation of how you solved Q2.4 is also required in the `pdf`. We do not accept handwritten scans in this assignment.

   • `eightpoint_norm.m`, `sevenpoint_norm.m`, `ransacF.m`, `genNovelViews.m` : requirements of Q2.1, Q2.2, Q2.3 and Q2.4.

   • Any other `.m` files which you wrote or used for the homework.

   • Remove data and temporary files: in `<NetID>.zip`, make sure you have removed the folder `data/`, `lib/` that we do not ask you to submit. When grading, the functions in `lib/` will be provided to your code, so you do not need to upload them.

   • `eightpoint_norm.m+, sevenpoint_norm.m, ransacF.m, genNovelViews.m` : requirements of Q2.1, Q2.2, Q2.3 and Q2.4.

   • Any other `.m` files which you wrote for the homework.

   • Remove data and temporary files: in `<NetID>.zip`, make sure you have removed the folder `data/`, `lib/` that we do not ask you to submit. When grading, the functions in `lib/` will be provided to your code, so you do not need to upload them.

   • Remove commented code that is part of your prior work. We will deduct marks for leftover code that we find making your work harder to read.

3. If you have questions, please post them on the BlackBoard first.

1 Theory Questions

Question 1.1 (5pts)

Suppose two cameras fixate on a point P (see Figure 1) in space such that their optical 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 F_{33} element of the fundamental matrix is zero.

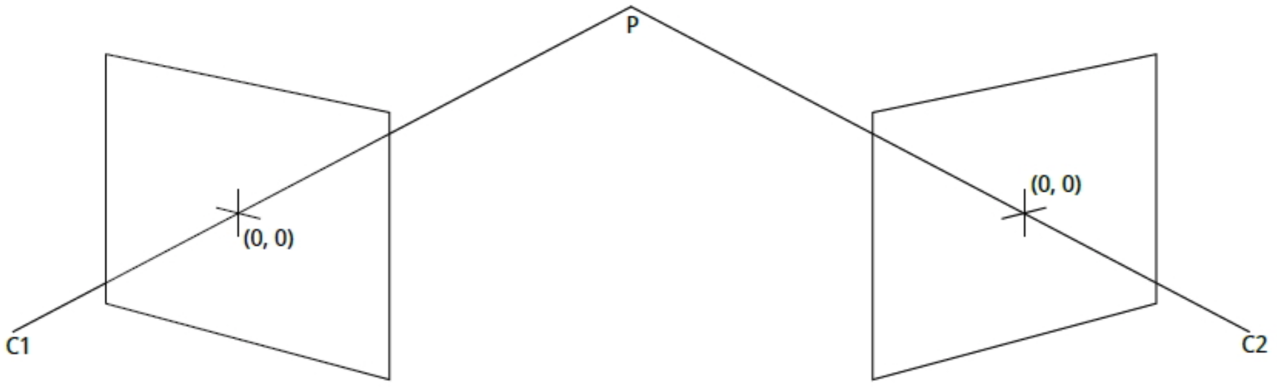


Figure 1: Figure for Q1.1. $C1$ and $C2$ are the optical centers. The principal axes intersect at point P.

Question 1.2 (10 pts)

Consider the case of two cameras viewing an object such that the second camera differs from the first by a pure translation that is parallel to the x-axis. Show that the epipolar lines in the two cameras are also parallel to the x-axis.

Question 1.3 (10 pts)

Show that the image of an object and the image of the same object viewed in a mirror are related by a skew-symmetric fundamental matrix, i.e., $F = -F^T$

Question 1.4 (10 pts)

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. Consider a point P in 3D, and its projection on camera 1 (p_1) and camera 2 (p_2). Assume that we know the homography H to map p_1 to p_2, i.e. $p_2 = Hp_1$.

- Assume that H is computed from a known plane where P is located, and the two cameras are separated by some translation and rotation (meaning that their centers are different). Given p_1, express the corresponding epipolar line in the second camera as a function of p_1, H, and e_2 (epipole in the second image).
- Assume that two cameras are separated by a pure rotation, and the H is computed from this relation. Is it possible to express the epipolar line similar to the previous case? If yes, express it. If not, explain the reason.

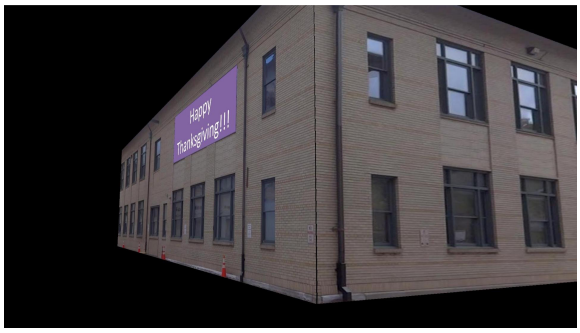
2 Implementation Questions

The main goal of this homework is to generate novel views of the lovely Smith Hall (the home of Computer Vision and CMU). You will be given two images of Smith Hall as shown in Figure 2a and Figure 2a. Then, with the techniques learned in class, we can recover the 3D scene and generate novel views of Smith Hall as shown in Figure 2c and Figure 2d. We will prepare you for this task by guiding you to implement some basic functions that would be necessary for this task. However, it will be up to you to design your method to complete this task given the basic functions.

Figure 2: Smith Hall viewed from different angles. Your final results will not have the purple text box.

(a) View 1 of Smith Hall

(b) View 2 of Smith Hall



(c) Novel view 1 of Smith Hall

(d) Novel view 2 of Smith Hall

In order to recover the 3D scene, we need to recover the relative positions and viewpoints of the two cameras. To recover this information, we start by computing the fundamental matrix F. Then, with F and the intrinsic parameters K, we can recover the essential matrix by $E = K^T F K$. This is assuming that the intrinsic parameters for both cameras are the same. Once we have the essential matrix, there exists a method[1] to recover the relative rotation R and translation t between the two cameras. With the relative rotation and translation, we can compute the camera matrices of the two cameras. Once we have the camera matrices, we can perform triangulation to location the point correspondences in 3D, as shown in Figure 3. In sum, computing the fundamental matrix is the first step to get this started. In this part you will begin by implementing the two different methods seen in class to estimate the fundamental matrix from point correspondences in two images.

Fundamental matrix estimation

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.

Question 2.1 (10 pts)

Submit a function with the following signature for this portion of the assignment:

```
F = eightpoint_norm(pts1, pts2, normalization_constant)
```

[1]More details here (`http://en.wikipedia.org/wiki/Essential_matrix`) and here (`lib/camera2.m`).

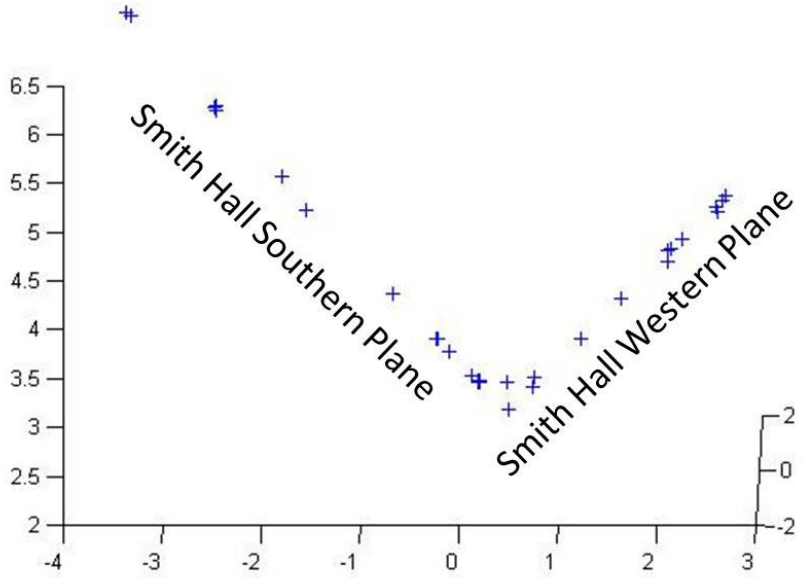


Figure 3: 3D reconstruction point cloud of Smith Hall

where `pts1` and `pts2` are $2 \times N$ matrices corresponding to the (x, y) coordinates (one point per column) of the points in the first and second image respectively. `normalization_constant` is a normalization constant to avoid numerical issues. Numerical issues are caused by the finite precision of floating point numbers in computers. Therefore, to avoid numerical problems, it is best practice to scale the point locations so that they are between [0,1]. For example, for `i1.jpg`, the width of the image is 1920, so the x-axis value will range from [1,1920]. We should normalize all x-axis values so that it ranges from [0, 1] instead. In practice, we usually set the normalization constant to the larger image dimension, i.e. the width of the image in this case. Mathematically, we are not doing anything different, but implementation wise this is important to get stable results. Finally, remember that the x-coordinate of a point in the image is its column entry, and y-coordinate is the row entry. Also note that eight-point is just a figurative name; your algorithm should use an over-determined system.

Run your function on the `pts1` and `pts2` stored in `lib/clean_correspondences.mat`. To visualize the correctness of your estimated $F_{8,clean}$, use the function `displayEpipolarF(i1, i2, F)` in `lib/displayEpipolarF.m`. This is a GUI that lets you select a point in one of the images and visualize the corresponding epipolar line in the other image (see Fig.4). You can find the two images of Smith Hall at `data/i1.jpg` and `data/i2.jpg`. Do the epipolar lines look reasonable? The correspondences used now are manually selected by humans, so there is very little noise in this set of correspondences.

However, in reality, point correspondences are found automatically, so there could potentially be lots of noise in the correspondences. Run your function on the pts1 and pts2 stored in lib/noisy correspondences.mat to compute $F_{8,noisy}$ and visualize the epipolar lines. How do they look? Comparing between using clean and noisy correspondences, which set of point correspondences have more reasonable epipolar lines? In your answer sheet: write your recovered $F_{8,clean}$ and $F_{8,noisy}$. Also print an output of `displayEpipolarF` similar to Figure 4 for both fundamental matrices $F_{8,clean}$ and $F_{8,noisy}$. Finally, in ≤ 2 sentences, please state which fundamental matrix estimate is more accurate, and why.

To deal with noisy point correspondences, we mentioned in class that we can use RANSAC (RANdom SAmple Consensus) to help remove noisy point correspondences. As mentioned in class, to enable RANSAC to find a correct F quicker, we want to estimate F with as few points

Figure 4: Snapshot of `displayEpipolarF`

as possible. Therefore, before we implement RANSAC, we want to implement the seven point algorithm first.

The Seven Point Algorithm

Question 2.2 (15 pts)

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, and outlined in Section 15.6 of Forsyth and Ponce. The function should have the signature:

```
F = sevenpoint_norm(pts1, pts2, normalization_constant)
```

where `pts1` and `pts2` are 2×7 matrices containing the correspondences, and F is a cell array of length either 1 or 3 containing Fundamental matrix/matrices. `normalization_constant` is the normalization constant to avoid numerical issues. Run your function on the first 7 correspondences of `lib/clean_correspondences.mat` to recover $F_{7,clean}$. Use `displayEpipolarF` to visualize $F_{7,clean}$ and pick the correct one. **In your answer sheet: write your recovered $F_{7,clean}$ and print an output of `displayEpipolarF`.**

Hints: you can use `roots`; the epipolar lines may not match exactly due to imperfectly selected correspondences.

Now that we have the seven point algorithm, we can proceed to RANSAC.

Computing F from Noisy Correspondences with RANSAC

Question 2.3 (10 pts)

To compute the fundamental matrix from noisy correspondences, we utilize both RANSAC and the seven point algorithm. In each iteration of RANSAC, 7 random point correspondences are chosen and provided to the seven point algorithm to compute a F. Then F is applied to each point correspondence (x_1, x_2) to check whether the geometric distance between point x_2 and epipolar line F_{x_1} is within a threshold (ideally, the point should be on the line, i.e. $x_2^T F x_1 = 0$). The point correspondences that pass the check are the "inliers". The goal of RANSAC is to try find a F with the most inliers. Please implement the function below:

```
[F, inliers] = ransacF(pts1, pts2, normalization_constant)
```

where `pts1` and `pts2` are $2 \times N$ matrices containing the correspondences, and F is a 3×3 fundamental matrix found by RANSAC. inliers is a $1 \times a$ vector which stores the indices of the a inliers of F.

Run RANSAC on the `pts1` and `pts2` stored in `lib/noisy_correspondences.mat` to compute $F_{7,RANSAC}$ and visualize the epipolar lines. How do they look? **In your answer sheet: write your recovered $F_{7,RANSAC}$. Also print an output of `displayEpipolarF` for $F_{7,RANSAC}$. Hint:** About half of the points in the data are noisy. So you may need to iterate many times (thousands of times) to get a good result.

Generating Novel Views of Smith Hall

Question 2.4 (30 pts)

Now that we have the basic tools for 3D reconstruction, you are ready to generate novel views of Smith Hall. You only need to focus on drawing the two main planes of Smith Hall. Here are a list of functions which might be useful for you.

1. Intrinsic parameters K are stored in `data/K.mat`.
2. `M2 = camera2(F, K1, K2, pts1, pts2)` in `lib/camera2.m`: assuming that the camera matrix M_1 for camera 1 is `K1 [I 0]`, it computes the camera matrix M_2 for camera 2. It requires the intrinsic parameters `K2` and the inlier point correspondences `pts1`, `pts2` which are $2 \times N$ matrices.
3. `P = triangulate(M1, pts1, M2, pts2)` in `lib/triangulate.m`: Given the two camera matrices M_1, M_2 and the inlier point correspondences, it computes the 3D location P of each point correspondence. `pts1`, `pts2` are the point correspondences stored in $2 \times N$ matrices. P is a $3 \times N$ matrix.
4. `frame = drawNovelView(smith_south_plane, smith_west_plane, M)`: Given the 3D plane parameters for the southern plane and the western plane of Smith Hall (as shown in Figure 5), it draws the scene viewed from camera matrix M. Figure 2c and Figure 2d are drawn with this function. `smith_south_plane`, `smith_west_plane` are 4×1 vectors which corresponds to [a b c d] for the plane equation $ax + by + cz + d = 0$.

Hints:

1. If a group of points belong to the same plane in 3D, there exists a homography to map the points between the two images. Can you use this fact to find planes in the scene?
2. There are two very obvious planes in Smith Hall. Take advantage of it.
3. Camera matrix $M \equiv K[Rt]$, and you can get the K, R, t of the original views using the provided functions above. Now your goal is to manipulate them and get novel Ms to generate novel views.

For this task, please use the point correspondences `data/noisy_correspondences.mat`. You should not need to do any more manual annotation. Your code should be fully automatic and not require a human in the loop.

Figure 5: The southern and western plane of Smith Hall.

Save your code in `genNovelView.m`. The script should run out-of-the-box. Dont forget to include all the `.m` files required by this script in your submission.

In your answer sheet, please explain clearly the steps you did to create the 2 novel views. Also, please show 2 novel views of Smith Hall. At least one of the views should be viewing Smith Hall from a higher angle like the one shown in Figure 2d.