

CS 231A Computer Vision (Winter 2016)

Problem Set 2

April 25th 2016 11:59pm

1 Fundamental Matrix (20 points)

In this question, you will explore some properties of fundamental matrix and derive a minimal parameterization for it.

- a Show that two 3×4 camera matrices M and M' can always be reduced to the following canonical forms by an appropriate projective transformation in 3D space, which is represented by a 4×4 matrix H . Here, we assume $e_3^T(-A'A^{-1}b+b') \neq 0$, where $e_3 = (0, 0, 1)^T$, $M = [A, b]$ and $M' = [A', b']$.

Note: You don't have to show the explicit form of H for the proof. **[10 points]**

Hint: The camera matrix has rank 3. Block matrix multiplication may be helpful. If you construct a projective transformation matrix H_0 that reduces M to \hat{M} , (i.e., $\hat{M} = MH_0$) can a H_1 be constructed such that not only does it not affect the reduction of M to \hat{M} (i.e., $\hat{M} = MH_0H_1$), but it also reduces M' to \hat{M}' ? (i.e., $\hat{M}' = M'H_0H_1$)

$$\hat{M} = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \end{bmatrix} \text{ and } \hat{M}' = \begin{bmatrix} a_{11} & a_{12} & a_{13} & b_1 \\ a_{21} & a_{22} & a_{23} & b_2 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

- b Given a 4×4 matrix H representing a projective transformation in 3D space, prove that the fundamental matrices corresponding to the two pairs of camera matrices (M, M') and $(MH, M'H)$ are the same. **[5 points]**

(Hint: Think about point correspondences)

- c Using the conclusions from (a) and (b), derive the fundamental matrix F of the camera pair (M, M') using $a_{11}, a_{12}, a_{13}, a_{21}, a_{22}, a_{23}, b_1, b_2$. Then use the fact that F is only defined up to a scale factor to construct a seven-parameter expression for F . *(Hint: The fundamental matrix corresponding to a pair of camera matrices $M = [I \mid 0]$ and $M' = [A \mid b]$ is equal to $[b]_{\times}A$.)* **[5 points]**

2 Fundamental Matrix Estimation From Point Correspondences (20 points)

This problem is concerned with the estimation of the fundamental matrix from point correspondences. Implement both the linear least-squares version of the eight-point algorithm and its normalized version. In both cases, enforce the rank-two constraint for the fundamental matrix via singular value decomposition.

The data for this problem is here: <http://cs231a.stanford.edu/hw/ps2/data.zip>. The zipped file contains two datasets. For each dataset, you will find in the corresponding directory the files

pt_2d_1.txt, pt_2d_2.txt
image1.jpg, image2.jpg

The first two of these files contain matching image points (in the following format: number n of points followed by n pairs of floating-point coordinates). The two remaining files are the images where the points were found.

You must turn in:

- a copy of your code, [15 points]
- For both methods applied to both datasets, the average distance between the points and the corresponding epipolar lines for each image, as well as a drawing similar to Figure 1 (with the selected corresponding points and their epipolar lines). [5 points]

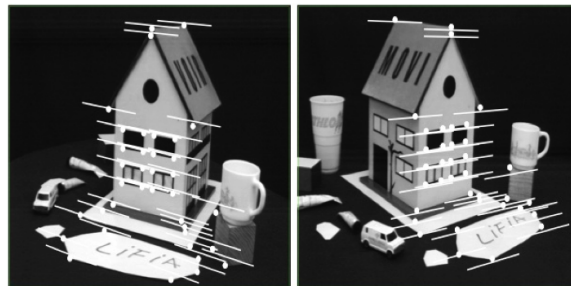


Figure 1: Example illustration, with epipolar lines shown in both images (Image courtesy Forsyth & Ponce)

3 Affine Triangulation using the Factorization Method (30 points)

In this question, you will explore the factorization method, initially presented by Tomasi and Kanade, for solving the affine structure from motion problem.

Begin by downloading http://cs231a.stanford.edu/hw/ps2/ps2_factor.zip and running the function `ps2q2.m`. Observe the point correspondences that have been selected in the two images.

- Now implement the factorization method presented in the lecture slides. You only need to modify the file `ps2q2.m`, and you only need to enter code in the location that says “YOUR CODE HERE.” [7 points]

- (b) Run your code and visualize the result by plotting the resulting 3d points; the `plot3` Matlab function may be helpful. We also recommend calling `axis equal` after making your plots so that the axes will be scaled appropriately.

Compare your result to the ground truth provided. The results should look identical, except for a scaling and rotation. Explain why this occurs. **[5 points]**

- (c) Report the 4 singular values from the SVD decomposition. Why are there 4 non-zero singular values? How many non-zero singular values would you expect to get in the idealized version of the method, and why? **[6 points]**
- (d) Change the file `readTextFiles.m` by setting `usesubset` to `true`, and re-run the code. The file will now only load a subset of the original correspondences. Compare your new results to the ground truth, and explain why they no longer appear similar. **[6 points]**
- (e) Report the new singular values, and compare them to the singular values that you found previously. Explain any major changes. **[6 points]**

4 Projective Triangulation in Structure From Motion (30 points)



Figure 2: The set of images used in this structure from motion reconstruction.

Structure from motion is inspired by our ability to learn about the 3D structure in the surrounding environment by moving through it. Given a sequence of images, we are able to simultaneously estimate both the 3D structure and the path the camera took. In this problem, you will implement significant parts of a structure from motion framework, estimating both R and T of the cameras, as well as generating the locations of points in 3D space. Recall that in the previous problem we triangulated points assuming affine transformations. However, in the actual structure from motion problem, we assume projective transformations. By doing this problem, you will learn how to solve this type of triangulation.

The starter code and data can be found at <http://cs231a.stanford.edu/hw/ps2/SFM.zip>

- (a) Given correspondences between pairs of images, we compute the respective Fundamental and Essential matrices. Given the Essential matrix, we must now compute the R and T between the two cameras. On pages 257-259 of HZ, the authors explain that this is possible as follows:

1. To compute R : Given the singular value decomposition $E = UDV^T$, we can rewrite

$$E = MQ \text{ where } M = UZU^T \text{ and } Q = UWV^T \text{ or } UW^TV^T, \text{ where } Z = \begin{bmatrix} 0 & 1 & 0 \\ -1 & 0 & 0 \\ 0 & 0 & 0 \end{bmatrix}$$

and $W = \begin{bmatrix} 0 & -1 & 0 \\ 1 & 0 & 0 \\ 0 & 0 & 1 \end{bmatrix}$. Note that this factorization of E only guarantees that Q is orthogonal. To find a rotation, we simply compute $R = Q$ if $\det Q > 0$ and $R = -Q$ if $\det Q < 0$.

2. To compute T : Given that $E = UDV^T$, T is simply either u_3 or $-u_3$, where u_3 is the third column vector of U .

Implement this in the function `computeRTFromE.m`. For now, verify that you get the correct four pairs of R and T . Give the four pairs of R and T for the sample matrix in `sfm.m`. You will submit your code for this problem in part e. **[5 points]**

- (b) In order to distinguish the correct R, T pair, we must first know how to find the 3D point given matching correspondences in different images. On page 312 of HZ, the authors explain a linear estimate (DLT) of this 3D point:

1. For each image i , we have $x_i = P_i X$, where X is the 3D point, x_i is the homogenous image coordinate of that point, and P_i is the projective camera matrix.
2. Formulate matrix

$$A = \begin{bmatrix} x_{1,1}p^{3\top} - p^{1\top} \\ x_{1,2}p^{3\top} - p^{2\top} \\ \vdots \\ x_{n,1}p^{3\top} - p^{1\top} \\ x_{n,2}p^{3\top} - p^{2\top} \end{bmatrix}$$

where $x_{i,1}$ and $x_{i,2}$ are the xy coordinates in image i and $p^{k\top}$ is the k -th row of P .

3. The 3D point is simply the vector in V corresponding to the smallest singular value of A .

Implement the linear estimate of this 3D point in `linearEstimate3D.m`. Submit your code and run the check in `sfm.m`. **[5 points]**

- (c) However, we can do better than linear estimates, but usually this falls under some iterative nonlinear optimization. To do this kind of optimization, we need some residual. A simple one is the reprojection error of the correspondences, which is computed as follows:

For each image i , given camera matrix P_i , the 3D point X , we compute $y = P_i X$, and find that the image coordinates

$$x'_i = \frac{1}{y_3} \begin{bmatrix} y_1 \\ y_2 \end{bmatrix}$$

Given the ground truth image coordinates x_i , the reprojection error e_i for image i is

$$e_i = x'_i - x_i$$

The Jacobian is written as follows:

$$J = \begin{bmatrix} \frac{\partial e_1}{\partial X_1} & \frac{\partial e_1}{\partial X_2} & \frac{\partial e_1}{\partial X_3} \\ \vdots & \vdots & \vdots \\ \frac{\partial e_m}{\partial X_1} & \frac{\partial e_m}{\partial X_2} & \frac{\partial e_m}{\partial X_3} \end{bmatrix}$$

Recall that each e_i is a vector of length two, so the whole Jacobian is a $2K \times 3$ matrix, where K is the number of cameras. Fill in `reprojectionError.m`, which computes the reprojection error and Jacobian for a 3D point and its list of images. Submit your code and run the check in `sfm.m`. [5 points]

- (d) Implement the Newton Step algorithm which finds an approximation to the 3D point that minimizes this reprojection error. Recall that the Newton Step needs a good initialization, which we have from our linear estimate in part b. Then we continually update X for ten iterations:

$$X = X - J^T(JJ^T)^{-1}e$$

where J and e are the Jacobian and error computed from the previous part.

Implement this Newton step algorithm in the `nonlinEstimate3D.m` function. Submit your code and run the check in `sfm.m`. [5 points]

- (e) Now finally, go back and distinguish the correct R, T pair from part a.
1. First, compute the location of the 3D point of each pair of correspondences given each R, T pair
 2. Given each R, T you will have to find the 3D point's location in that R, T frame. The correct R, T pair is the one for which the most 3D points have positive depth (z-coordinate) in all camera frames. Page 259 of HZ has an excellent diagram and explanation of this.

Submit your updated code here. After doing so, we now have a robust method to triangulate 3D points given correspondences. Submit the final plot of the reconstruction. Hopefully, you can see a point cloud that looks like the frontal part of the statue in the above sequence of images: Note that to better see the point cloud, try rotating or wiggling the plot. You

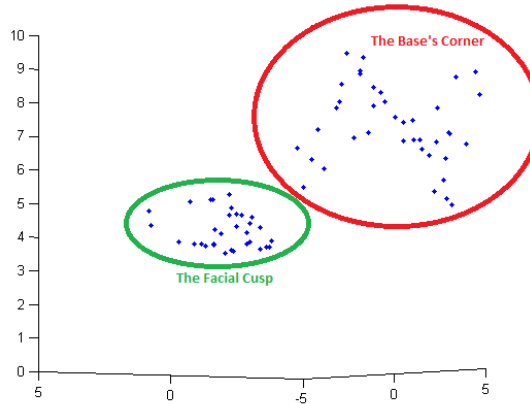


Figure 3: The final reconstruction our solution renders.

should be able to see the corner of the base and the cusp that forms the facial part of the statue. [10 points]