

Computer Vision I
CSE 252a, Fall 2012
David Kriegman

Name:
Student ID:
Email:

Assignment #3

(Due date: 11/20/2012)

Problem 1. Epipolar Geometry [2pts]

Consider two cameras whose image planes are the $z=1$ plane, and whose focal points are at $(-20, 0, 0)$ and $(20, 0, 0)$. We'll call a point in the first camera (x, y) , and a point in the second camera (u, v) . Points in each camera are relative to the camera center. So, for example if $(x, y) = (0, 0)$, this is really the point $(-20, 0, 1)$ in world coordinates, while if $(u, v) = (0, 0)$ this is the point $(20, 0, 1)$.



Figure 1: Problem 1 Setup

- Suppose the points $(x, y) = (8, 8)$ is matched with disparity of 7 to the point $(u, v) = (1, 8)$. What is the 3D location of this point?
- Consider points that lie on the line $x + z = 0, y = 0$. Use the same stereo set up as before. Write an analytic expression giving the disparity of a point on this line after it projects onto the two images, as a function of its position in the right image. So your expression should only involve the variables u and d (for disparity). Your expression only needs to be valid for points on the line that are in front of the cameras, i.e. with $z > 1$

Problem 2. Epipolar Rectification [3pts]

In stereo vision, image rectification is a common preprocessing step to simplify the problem of finding matching points between images. The goal is to warp image views such that the epipolar lines are horizontal scan lines of the input images. Suppose that we have captured two images IA and IB from identical calibrated cameras separated by a rigid transformation

$${}_A^S T = \begin{bmatrix} R & t \\ 0^T & 1 \end{bmatrix}$$

Without loss of generality assume that camera A's optical center is positioned at the origin and that its optical axis is in the direction of the z -axis.

From the lecture, a rectifying transform for each image should map the epipole to the point infinitely far away in the horizontal direction $H_A e_A = H_B e_B = [1, 0, 0]^T$. Consider the following special cases:

- Pure horizontal translation $t = [t_x, 0, 0]^T, R = I$
- Pure translation orthogonal to the optical axis $t = [t_x, t_y, 0]^T, R = I$
- Pure translation along the optical axis $t = [0, 0, t_z]^T, R = I$
- Pure rotation $t = [0, 0, 0]^T, R$ is an arbitrary rotation matrix.

For each of these cases, determine whether or not epipolar rectification is possible. Include the following information for each case:

- The epipoles e_A and e_B
- The equation of the epipolar line l_B in I_B corresponding to the point $[x_A, y_A, 1]^T$ in I_A (if one exists)
- A plausible solution to the rectifying transforms H_A and H_B (if one exists) that attempts to minimize distortion (is as close as possible to a 2D rigid transformation). Note that the above 4 cases are special cases; a simple solution should become apparent by looking at the epipolar lines.

One or more of the above rigid transformations may be a degenerate case where rectification is not possible or epipolar geometry does not apply. If so, explain why.

Problem 3. NCC [2pts]

Show that maximizing the NCC $\sum_{i,j} \tilde{W}_1(i,j) \cdot \tilde{W}_2(i,j)$ is equivalent to minimizing the NSSD $\sum_{i,j} |\tilde{W}_1(i,j) - \tilde{W}_2(i,j)|^2$.

Hint: Express c_{NCC} and c_{NSSD} in terms of the vectors $\tilde{w}_1 = \tilde{W}_1(:)$ and $\tilde{w}_2 = \tilde{W}_2(:)$. Thus, $c_{NCC} = \tilde{w}_1^T \tilde{w}_2$ and $c_{NSSD} = (\tilde{w}_1 - \tilde{w}_2)^T (\tilde{w}_1 - \tilde{w}_2)$

$\tilde{W} = \frac{W - \bar{W}}{\sqrt{(\sum_{k,l} (W(k,l) - \bar{W})^2)}}$ is a mean-shifted and normalized version of the window. N refers to the number of pixels in each window.

Problem 4. 3-D Reconstruction [10 Pts]

This is a programming assignment, which should be done in Matlab as many of the necessary numerical routines are readily available. All data necessary for this assignment is available on the course web page.

Corner Detection

Implement a corner detector as described in the lecture. Recall that the corner detector also computes the minimum eigenvalue of the matrix:

$$A^T A = \begin{pmatrix} \sum I_x^2 & \sum I_x I_y \\ \sum I_y I_x & \sum I_y^2 \end{pmatrix}$$

and selects corner points that are local maxima and above some threshold. A popular way of avoiding comparing the eigenvalues of $A^T A$ directly is to use the following formula:

$$R = \lambda_1 \lambda_2 - k(\lambda_1 + \lambda_2)^2$$

where k is a thresholding factor and R is the resulting weight factor.

- Write a function with the following signature:

`[x,y] = getInterestPoints(img, windowSize, numPoints,K)`. This function should detect corners with the specified parameters and return their xy coordinates (x y should be column vectors). You should return only the best `numPoints` of R . Then show images of detected corner points on the following images using the matlab command `plot(x,y,'o')` for both $K = 0.001$ and $K = 0.01$.

warrior01, windowSize = 11x11

dino01, windowSize = 15x15

matrix01, windowSize = 15x15.

Does this algorithm work well for these test images? If not, why?

Here are a few implementation tips:

- Look into alternative ways of finding the eigenvalues besides using `eig` or `svd` to speed up your algorithm.
- Some matlab commands that may be useful: `imfilter`, `imregionalmax`
- You'll want to remove any points along the image border to avoid points that may not be available when using multiple images.

The Seven Point Algorithm

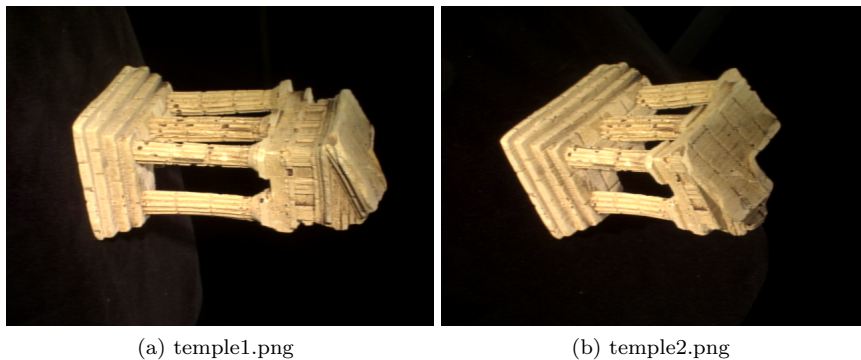


Figure 2: Temple images

In this section of the assignment you will be calculating the fundamental matrix F . The fundamental matrix has seven degrees of freedom, therefore it is possible to find it using only seven correspondences by solving a polynomial equation. In this problem you will write a function to calculate the fundamental matrix $\mathbf{F}$ using two sets of points. You will find the seven point algorithm outlined in Section 7.1 7.2 and 8.1 of Forsyth and Ponce and p. 350 in Szeliski¹.

- Implement a function to compute $\mathbf{F}$ using the seven point algorithm. This function should return either 1 or 3 matrices for $\mathbf{F}$. Test this function on points in `corresp_7pt.mat` contained in `temple.zip`. Then, use `displayEpipolarF` to visualize $\mathbf{F}$ and pick the correct one. The temple images are also contained in the zip-file.

Here are a few implementation tips:

- You'll want to pass a normalization factor into this function to normalize the point values to $[0,1]$, using the max of the image dimensions will suffice.
- You'll also need to 'unnormalize' $\mathbf{F}$. Remember if $\hat{x} = Tx$ then $F = T^T \hat{F} T$
- You may want to look into the function `roots` to solve your polynomial.

¹Both the seven point and the eight point algorithm are described here. These two algorithms are very similar.

Automatic Computation of F

In many real-world applications manually finding correspondences may be impractical. There are a number of methods to help estimate the epipolar geometry between images. In this problem you will implement a popular algorithm for this called RANSAC (**R**andom **S**ampling **C**onsensus). This algorithm was outline in lecture and can be found in the text Section 10.4 of Forsyth and Ponce and p. 318 in Szeliski. We will provide a function that will determine candidate correspondences. This function finds correspondences using the smallest Sum of Absolute Differences (SAD) in feature windows. It returns the features that matched the best when comparing in either direction. Please double check to ensure that `getInterestPoints` is in your current directory or in your matlab path for the provided `getCandidateCorrespondences` to work properly.

- c. Write a function that implements RANSAC to find the best F after a number of iterations (several hundred) on the images in temple.zip. This function should use `getCandidateCorrespondences` as well as your `getInterestPoints` and seven point algorithm function. Then use `displayEpipolarF` to visualize the resulting F .

What to turn in:

Submit a hardcopy on the due date with the following:

- Your code for: The corner detector, seven point algorithm and RANSAC.
- All three of your figures for warrior01, dino01, matrix01 with your corner detector results and any modifications to the window size and/or thresholds you used.
- A print out of your manually chosen F and F chosen by RANSAC
- Figures of `displayEpipolarF` of both F matrices

In addition, please email to jmerkow@ucsd.edu a copy of your code and figures. Please send as a zip or tar file. In the subject line, please put the string CSE252AHW3.