CSx495 — Computer Vision
Problem Set 3: Geometry

The past several lectures have dealt with the geometry of imaging. In this project you will epxlore *calibrating* a camera with respect to 3D world coordinates as well as esitmating the relationship between two camera views.

As a reminder as to what you hand in: A Zip file that has

1. Images (either as JPG or PNG or some other easy to recognize format) clearly labeled using the convention PS<number>–<question number>–<question sub>–counter.jpg
2. Code you used for each question. It should be clear how to run your code to generate the results. Code should be in different folders for each main part with names like PS1-1-code. For some parts – especially if using Matlab – the entire code might be just a few lines.
3. Finally a PDF file that shows all the results you need for the problem set. This will include the images appropriately labeled so it is clear which section they are for and the small number of written responses necessary to answer some of the questions. Also, for each main section, if it is not obvious how to run your code please provide brief but clear instructions. **If there is no Zip file you will lose at least 50%!**

This project uses files stored in the directory `http://www.cc.gatech.edu/~afb/classes/CS4495-Fall2013/ProblemSets/PS3`.

1 Calibration

The files pts2d-pic_a.txt and pts3d.txt are a list of twenty 2D and 3D points of the image pic_a.jpg The goal is to compute the projection matrix that goes from world 3D coordinates to 2D image coordinates. Recall that using homogeneous coordinates the equation is:

$$\begin{bmatrix} u \\ v \\ 1 \end{bmatrix} \simeq \begin{bmatrix} s*u \\ s*v \\ s \end{bmatrix} = \begin{bmatrix} m_{1,1} & m_{1,2} & m_{1,3} & m_{1,4} \\ m_{2,1} & m_{2,2} & m_{2,3} & m_{2,4} \\ m_{3,1} & m_{3,2} & m_{3,3} & m_{3,4} \end{bmatrix} \begin{bmatrix} X \\ Y \\ Z \\ 1 \end{bmatrix}$$

Recall you solve for the 3x4 matrix M using either SVD to solve the homogeneous version of the equations or by setting $m_{4,4}$ to 1 and then using the a normal least squares method. Remember that M is only known up to a scale factor.

To make sure that your code is correct, we are going to give you a set of "normalized points" in the files `./pts2d-norm-pic_a.txt` and `./pts3d-norm.txt`. If you solve for M using all the points you should get a matrix that is a scaled equivalent of the following:

$$M_{\text{normA}} = \begin{bmatrix} -0.4583 & 0.2947 & 0.0139 & -0.0040 \\ 0.0509 & 0.0546 & 0.5410 & 0.0524 \\ -0.1090 & -0.1784 & 0.0443 & -0.5968 \end{bmatrix}$$

For example, this matrix will take the last normalized 3D point which is $< 1.2323, 1.4421, 0.4506, 1.0 >$ and will project it to the $< u, v >$ of $< 0.1419, -0.4518 >$ where we converted the homogeneous 2D point $< us, vs, s >$ to its inhomogeneous version by dividing by s (i.e. the real transformed pixel in the image).

1.1 Create the least squares function that will solve for the 3x4 matrix M_{normA} given the normalized 2D and 3D lists, namely `./pts2d-norm-pic_a.txt` and `./pts3d-norm.txt`. Test it on the normalized 3D points by multiplying those points by your M matrix and comparing the resulting the normalized 2D points to the normalized 2D points given in the file. Remember to divide by the homogeneous value to get an inhomogeneous point. You can do the comparison by checking the *residual* between the predicted location of each test point using your equation and the actual location given by the 2D input data. The residual is just the distance (square root of the sum of squared differences in u and v).

Output: code that does the solving, the matrix M you recovered from the normalized points, the $< u, v >$ projection of the last point given your M matrix, and the residual between that projected location and the actual one given in the file.

Now you are ready to calibrate the cameras. Using the 3D and 2D point lists for the image, we're going to compute the camera projection matrix. To understand the effects of overconstraining the system, you're going to try using sets of 8, 12 and 16 points and then look at the residuals. To debug your code you can used the normalized set from above but for the actual question you'll need to use the `./pts2d-pic_b.txt` and `./pts3d.txt`

1.2 For the three point set sizes k of 8, 12, and 16, repeat 10 times:

1. Randomly choose k points from the 2D list and their corresponding points in the 3D list.
2. Compute the projection matrix M on the chosen points.
3. Pick 4 points not in your set of k and compute the average residual.
4. Save the M that gives the lowest residual.

Output: code that does the computation, and the average residual for each trial of each k (so that would be 10 x 3 = 30 numbers). Explain any difference you see between the results for the different k.

Output: Best M

Finally we can solve for the camera center in the world. Let us define M as being made up of a 3x3 we'll call Q and a 4_{th} column will call $\mathbf{m_4}$:

$$M = [\mathbf{Q}|\mathbf{m_4}]$$

From class we said that the center of the camera C could be found by:

$$C = -Q^{-1}\mathbf{m_4}$$

To debug your code: If you use you the normalized 3D points to get the M_{normA} given above you would get a camera center of:

$$C_{\text{normA}} =< -1.5125, -2.3515, 0.2826 >$$

1.3 Given the best M from the last part, compute C.

Output: code that does the computation, and the location of the camera in real 3D world coordinates.

2 Fundamental Matrix Estimation

We now wish to estimate the mapping of points in one image to lines in another by means of the fundamental matrix. This will require you to use similar methods to those in Problem 1. We will make use of the corresponding point locations listed in pts2d-pic_a.txt and pts2d-pic_b.txt.

Recall that the definition of the Fundamental Matrix is

$$\begin{bmatrix} u' & v' & 1 \end{bmatrix} \begin{bmatrix} f_{1,1} & f_{1,2} & f_{1,3} \\ f_{2,1} & f_{2,2} & f_{2,3} \\ f_{3,1} & f_{3,2} & f_{3,3} \end{bmatrix} \begin{bmatrix} u \\ v \\ 1 \end{bmatrix} = 0$$

Given corresponding points you get one equation per point pair. With 8 or more points you can solve this. With more points (such as the 20 in the files) you solve using the the same least squares method as in problem 1 above.

2.1 Create the least squares function that will solve for the 3x3 matrix $\tilde{F}$ that satisfies the epipolar constraints defined by the sets of corresponding points. Solve this function to create your least squares estimate of the 3x3 transform $\tilde{F}$.

Output: code that does the solving. The matrix $\tilde{F}$ generated from your least squares function.

2.2 The linear squares estimate of $\tilde{F}$ is full rank; however, the fundamental matrix is a rank 2 matrix. As such we must reduce its rank. In order to do this we can decompose $\tilde{F}$ using singular value decomposition into the matrices $U\Sigma V^T = \tilde{F}$. We can then estimate a rank 2 matrix by setting the smallest singular value in Σ to zero thus generating Σ'. The fundamental matrix is then easily calculated as $F = U\Sigma' V^T$. Use the SVD function to do, well, the SVD. Duh.

Output: Code and fundamental matrix F.

2.3 Now you can use your matrix F to estimate an epipolar line l_b in image 'b' corresponding to point p_a in image 'a':

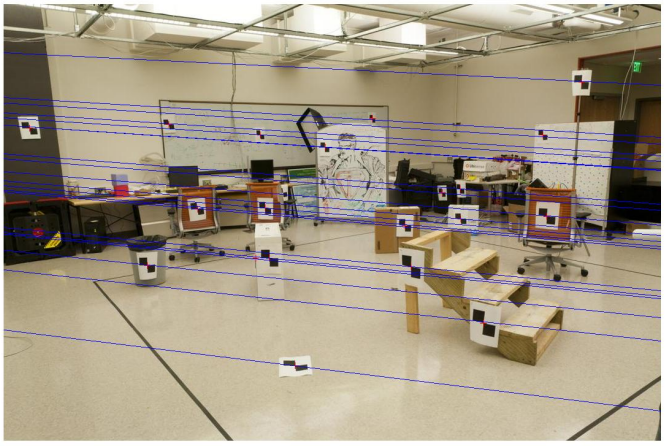
$$l_b = Fp_a$$

Similarly, epipolar lines in image a corresponding to points in image b are related by the transpose of F.

[Below is one way to draw the epipolar lines. The TAs may post additional ways. The key is to be able to take the projective geometry form of the line and draw it in the image.]

The resulting lines l_i defined in homogeneous coordinates can not be drawn using standard line functions, which take as input two points. In order to use such functions, we can find the intersection of a given line l_i with the image boundaries. If we define the line l_L to be the line corresponding to the left hand side of the image and l_R to be the line corresponding to the right hand side of the image, we can find the point $P_{i,L} = l_i \times l_L$ and $P_{i,R} = l_i \times l_R$. We can now plot the line running through the points $P_{i,L}, P_{i,R}$. However, we must first have the equations for l_L and l_R making use of the point-line duality, we know that $l_L = P_{UL} \times P_{BL}$. Where P_{UL} is the point defining the upper left-hand corner of the image and P_{BL} is the bottom left-hand corner of the image.

An example of such an image is:



Output: Code to perform the estimation and line drawing. Images with the estimated epipolar lines drawn on them.

If you look at the results of the last section the epipolar lines are close, but not perfect. The problem is that the offset and scale of the points is large and biased compared to some of the constants. To fix this, we are going to normalize the points.

In the 2D case this normalization is really easy. We want to construct transformations that make the mean of the points zero and, optionally, scale them so that their magnitude is about 1.0 or some other not too large number.

$$\begin{bmatrix} u' \\ v' \\ 1 \end{bmatrix} = \begin{bmatrix} s & 0 & 0 \\ 0 & s & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} 1 & 0 & -c_u \\ 0 & 1 & -c_v \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} u \\ v \\ 1 \end{bmatrix}$$

The transform matrix T is the product of the scale and offset matrices. The c_u, c_v is just the mean. To compute a scale s you could estimate the standard deviation after substracting the means. Or you could find the maximum absolute value. Then the scale factor s would be the reciprocal of whatever estimate of the scale you are using.

2.4 Create a two matrics T_a and T_b for the set of points defined in the files `./pts2d-pic_a.txt` and `./pts2d-pic_b.txt` respectively. Use these matrices to transform the two sets of points. Then use these normalized points to create a new Fundamental matrix $\hat{F}$. Compute it as above including making the smaller singular value zero.

Output: The matrixes T_a, T_b and $\hat{F}$

Finally you can create a better F by:

$$F = T_b^T \hat{F} T_a$$

2.5 Using the new F redraw the epipolar lines of 2.3. They should be better.

Output: The new F and the images with the "better" epipolar lines drawn.